Tivoli System Automation for z/OS

IBM

Version 3 Release 2

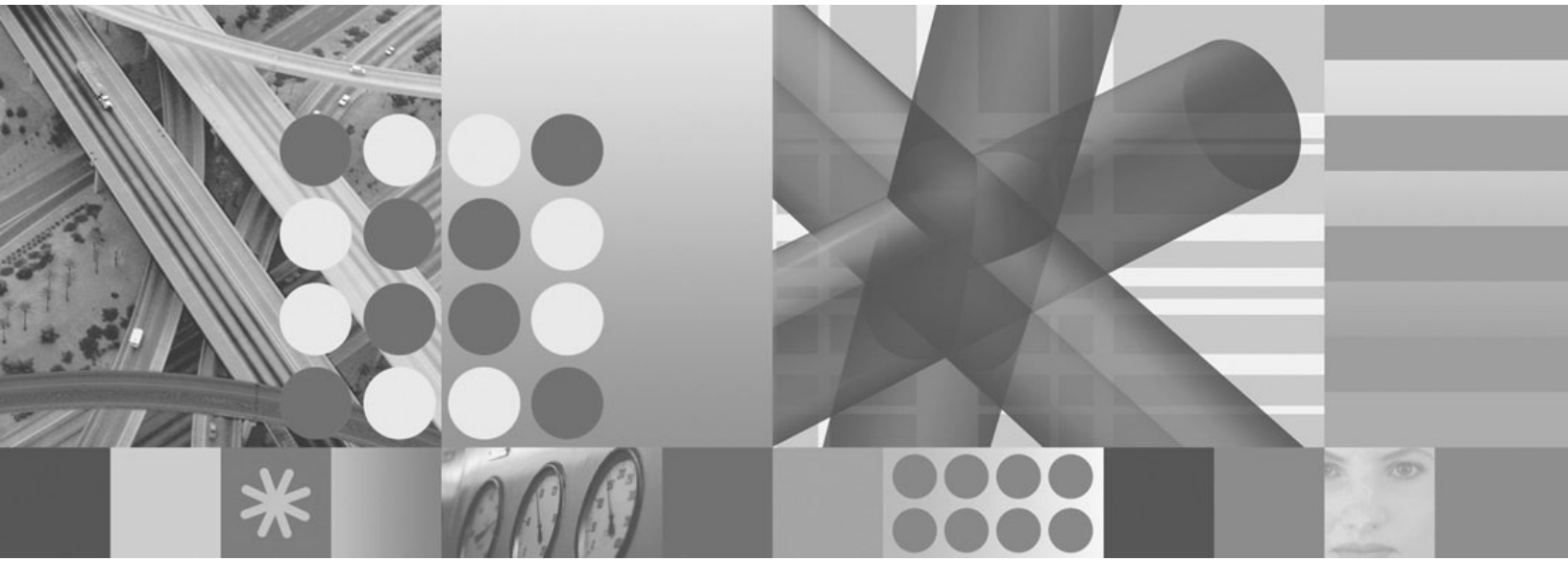
Customizing and Programming

SC33-8260-05

Tivoli® System Automation for z/OS

IBM

Version 3 Release 2



Customizing and Programming

SC33-8260-05

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

This edition applies to IBM Tivoli System Automation for z/OS (Program Number 5698-SA3) Version 3, Release 2, an IBM licensed program, and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

IBM welcomes your comments. A form for readers' comments appears at the back of this publication. If the form has been removed, address your comments to:

IBM Deutschland Entwicklung GmbH
Department 3248
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

If you prefer to send comments electronically, use one of the following methods:

FAX (Germany): 07031 + 16-3456
FAX (Other Countries): (+49)+7031-16-3456
Internet: s390id@de.ibm.com

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

Contents

Figures

Tables

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 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
USA

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 Deutschland Entwicklung GmbH
Department 3248
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

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 may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM 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.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming Interface Information

This publication primarily documents information that is *not* intended to be used as a Programming Interface of System Automation for z/OS.

This publication also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of System Automation for z/OS.

This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

Programming Interface information

This section contains Programming Interface Information.

End of Programming Interface information

Trademarks

The following terms are trademarks or service marks of the IBM Corporation in the United States or other countries, or both:

CICS®
DB2®
IBM
MVS™
OS/390®
PR/SM™
RACF®
RMF™
Tivoli®
VTAM®
z/OS
zSeries®
CICSPlex
GDPS
IMS™
NetView
Parallel Sysplex
Processor Resource/Systems Manager™
REXX
SecureWay
Tivoli Enterprise Console®
WebSphere®
z/VM®

The following terms are trademarks of other companies:

- Linux® is a registered trademark of Linus Torvalds.
- UNIX® is a registered trademark of The Open Group in the United States and other countries.

Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS™ enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Vol I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer/Library Server versions of z/OS books in the Internet library at:

`http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`

About This Book

This book describes how to adapt your completed standard installation of IBM Tivoli System Automation for z/OS (SA z/OS) as described in *IBM Tivoli System Automation for z/OS Planning and Installation* to your environment. This book contains information on how to add new applications to automation and how to write your own automation procedures. It also contains information about how to add new messages for automated applications.

Who Should Use This Book

This book is primarily intended for automation programmers responsible for:

- Customizing system automation and the operations environment
- Developing automation procedures and other operations capabilities

Prerequisites

Throughout this book, it is expected that readers will be familiar with System Automation for z/OS and the following documentation:

- *IBM Tivoli System Automation for z/OS Operator's Commands*
- *IBM Tivoli System Automation for z/OS Programmer's Reference*
- *IBM Tivoli System Automation for z/OS Defining Automation Policy*

Where to Find More Information

The System Automation for z/OS Library

The following table shows the information units in the System Automation for z/OS library:

Table 1. System Automation for z/OS Library

Title	Order Number
IBM Tivoli System Automation for z/OS Planning and Installation	SC33-8261
IBM Tivoli System Automation for z/OS Customizing and Programming	SC33-8260
IBM Tivoli System Automation for z/OS Defining Automation Policy	SC33-8262
IBM Tivoli System Automation for z/OS User's Guide	SC33-8263
IBM Tivoli System Automation for z/OS Messages and Codes	SC33-8264
IBM Tivoli System Automation for z/OS Operator's Commands	SC33-8265
IBM Tivoli System Automation for z/OS Programmer's Reference	SC33-8266
IBM Tivoli System Automation for z/OS CICS Automation Programmer's Reference and Operator's Guide	SC33-8267
IBM Tivoli System Automation for z/OS IMS Automation Programmer's Reference and Operator's Guide	SC33-8268
IBM Tivoli System Automation for z/OS TWS Automation Programmer's Reference and Operator's Guide	SC23-8269
IBM Tivoli System Automation for z/OS End-to-End Automation Adapter	SC33-8271

Table 1. System Automation for z/OS Library (continued)

Title	Order Number
IBM Tivoli System Automation for z/OS Monitoring Agent Configuration and User's Guide	SC33-8337

The System Automation for z/OS books are also available on CD-ROM as part of the following collection kit:

IBM Online Library z/OS Software Products Collection (SK3T-4270)

SA z/OS Home Page

For the latest news on SA z/OS, visit the SA z/OS home page at http://www.ibm.com/servers/eserver/zseries/software/sa

Related Product Information

You can find books in related product libraries that may be useful for support of the SA z/OS base program by visiting the z/OS Internet Library at http://www.ibm.com/servers/eserver/zseries/zos/bkserv/

Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most of the IBM messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can use LookAt from these locations to find IBM message explanations for z/OS elements and features, z/VM, VSE/ESA™, and Clusters for AIX® and Linux:

- The Internet. You can access IBM message explanations directly from the LookAt Web site at http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/.
- Your z/OS TSO/E host system. You can install code on your z/OS or z/OS.e systems to access IBM message explanations using LookAt from a TSO/E command line (for example: TSO/E prompt, ISPF, or z/OS UNIX System Services).
- Your Microsoft® Windows® workstation. You can install LookAt directly from the *z/OS Collection* (SK3T-4269) or the *z/OS and Software Products DVD Collection* (SK3T4271) and use it from the resulting Windows graphical user interface (GUI). The command prompt (also known as the DOS > command line) version can still be used from the directory in which you install the Windows version of LookAt.
- Your wireless handheld device. You can use the LookAt Mobile Edition from http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookatm.html with a handheld device that has wireless access and an Internet browser (for example: Internet Explorer for Pocket PCs, Blazer or Eudora for Palm OS, or Opera for Linux handheld devices).

You can obtain code to install LookAt on your host system or Microsoft Windows workstation from:

- A CD-ROM in the *z/OS Collection* (SK3T-4269).
- The *z/OS and Software Products DVD Collection* (SK3T4271).

- The LookAt Web site (click **Download** and then select the platform, release, collection, and location that suit your needs). More information is available in the LOOKAT.ME files available during the download process.

Summary of Changes for SC33-8260-05

This document contains information previously presented in System Automation for z/OS V3R1.0 Customizing and Programming, SC33-8260-04.

New Information

Availability and recovery time reporting
SA z/OS introduces support to assist you in billing users or reporting reliability of your critical applications or the software that those applications are dependent on. See Chapter 6, "Availability and Recovery Time Reporting," on page 73.

WTOR Processing
The processing of WTORs has been changed. See Chapter 12, "WTOR Processing," on page 149 for more details.

User exit routines
The following user exit routines are newly documented:

- "AOFEXI05" on page 156
- "AOFEXI06" on page 156
- "AOFEXX04" on page 158
- "AOFEXC17" on page 168

Changed Information

Assist Mode Facility
The assist mode facility has been changed. For more information, see "The Assist Mode Facility" on page 16 and "Using Assist Mode to Test Automation Procedures" on page 17.

IMS Transaction Recovery
The IMS transaction recovery function has been changed. For more information, see "IMS Transaction Recovery" on page 174.

Restricted message IDs
The list of restricted message IDs has been updated. See "Restricted Message IDs" on page 266.

Moved Information

The chapter "Exception-Based Monitoring with OMEGAMON" has been incorporated into Chapter 4, "How to Monitor Applications," on page 41.

To help locate the information in Chapter 14, "Automation Routines," on page 171 more easily, the automation routines are now provided alphabetically. The descriptions of the automation processing that these routines can be used for are now given at the start of the chapter.

Deleted Information

The following sections have been deleted:

- "Programming Additional SA z/OS Automation Procedures" in Chapter 2, "How to Create Automation Procedures," on page 5

- "Resolving a System Log Failure" and "Enabling System Log Failure Recovery" in Chapter 10, "How to Enable Sysplex Automation," on page 109
- "JES2 Shutdown Processing" and "JES3 Dump Processing" in Chapter 14, "Automation Routines," on page 171
- "Restricted Message IDs" in Appendix C, "Message Automation," on page 263. See the customization dialog online help for a complete list of restricted message IDs.

The following Advanced Automation Options have been deleted:

- AOF3WTIME
- AOFCTLOPT
- AOFIMSCMDMSG
- AOFMOVOPT
- AOFQUICKWTOR
- AOFRELOADOPT

The following global variables have been removed:

- AOFSHUTCHK
- AOFSHUTOVERRIDE
- AOFSHUTSCOPE

The AOFEXX01 user exit routine has been deleted.

The following automation routines have been deleted:

- EVERCMRC
- EVERSCMI
- EVEECMSI
- HASP099

As a result of enhancements to the *IMS sample policy and further integration of IMS automation into SA z/OS, details about the following automation routines have been deleted:

IMS Region Abend Recovery
EVIER000
EVIER001

IMS Dependent Region Processing
EVIES002
EVIES003

IMS MSC Link Recovery
EVIEY00S

IMS OLDS Recovery
EVIECO05
EVIEY00S

IMS RECON Recovery
EVIECR04

IMS Startup
EVIDISCQ
EVIEI00A
EVIEI00C
EVIEI20B

EVIEI200

IMS Shutdown
EVIET006

IMS Transaction Recovery
EVIEY00S

IMS XRF Processing
EVIAVM06
EVIEI005
EVIEI008
EVIEI009
EVIEI00D
EVIEI00F
EVIEI00G
EVIEO000
EVIEO001
EVIEO002
EVIEO006
EVIEO007
EVIEO008
EVIEO010
EVIET00E

You may notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Chapter 1. How to Add a New Application to Automation

This chapter describes the steps that are required to automate and monitor a new application by SA z/OS.

The main tasks involved in extending automation include:

- Adding or changing values via the SA z/OS customization dialog
- Building a new automation control file
- Adding or changing entries in the message processing facility (MPF) message list and the NetView automation table.
- Adding new resources to the status display facility (SDF)
- Reloading the changed files and tables, such as the MPF list, NetView automation table, and automation control file, to enable the new or changed automation

Note that messages that have been defined for the automation are automatically added to the NetView Automation Table and MPFLSTSA member. For more details see Chapter 3, "How to Add a Message to Automation," on page 21.

Step 1: Define an Application Policy Object

To add a new application to SA z/OS, you must create and define a new Application policy object using the SA z/OS customization dialog. With the customization dialog, you also define how the new application should be automated by SA z/OS, for example:

- Setting automation flags for the application
- Specifying startup or shutdown commands for the application
- Linking the application into an application group

How to do this is described in detail in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Step 2: Define Outstanding Reply Processing

SA z/OS keeps track of all outstanding Write-to-Operator Replies (WTORs) that it receives if it does not reply to them immediately. Because some applications may have more than one WTOR at the same time, and not all WTORs are equally important, they are classified accordingly. For more details refer to Chapter 12, "WTOR Processing," on page 149.

Step 3: Build New System Operations Configuration Files

When you finish defining the application in the customization dialog, build the new system operations configuration files (automation control file, automation manager configuration file, NetView Automation Tables, and the MPFLSTSA member) from the updated policy database. See *IBM Tivoli System Automation for z/OS Defining Automation Policy* for more information.

After you have completed this step and "Step 7: Reload MPF List and Automation Configuration Files" on page 3, the application is known to SA z/OS and can therefore be automated according to the policy that was defined in "Step 1: Define an Application Policy Object" on page 1.

For advanced application automation, you should consider completing some or all of the following steps.

Step 4: (Optional) Code Entries for Application Messages in the MPF List

If necessary, code your entries for the application startup, abend, and shutdown messages in the MPF list, specifying the AUTO(YES) parameter. This step is optional. If the default is already AUTO(YES) for the messages, bypass this step.

You can use the MPFLSTSA member, which contains all messages that are relevant to SA z/OS, as a basis for your own messages.

If you are automating a message, you probably also want to suppress the message from appearing on operator consoles. To mark a message for suppression, code SUP(YES) in the MPF list entry for the message.

For more information on coding MPF list entries, see *z/OS MVS Initialization and Tuning Reference*.

Step 5: Decide Where to Forward Subsystem Information To

You can use the following values in the **Inform List** field of the APPLICATION INFO policy item to determine where to forward subsystem information to:

SDF This registers the application with the Status Display Facility (SDF), allowing status changes to be sent to SDF. You must also update the SDF tree structure and SDF panels with information about the new application. Refer to Appendix B, "Customizing the Status Display Facility," on page 247 for more details.

NMC This registers the application with the NetView Management Console (NMC), allowing status changes to be sent to NMC. You must also perform "Step 6: Enable the Application for the SA z/OS Graphical Interface" on page 3.

IOM This allows alerts for the application to be sent to System Automation for Integrated Operations Management (SA IOM) for notification processing. Refer to Chapter 5, "Alert-Based Notification," on page 67 for more information.

SMF This enables SA z/OS to collect and record job-related information, and write System Management Facility (SMF) records at specific events in the lifetime of the application. Refer to Chapter 6, "Availability and Recovery Time Reporting," on page 73 for more information.

NONE
The application is not registered at all. This prevents inheritance from the system's default definitions (SDF) or application default definitions (ADF).

blank The value is obtained from the system's default definitions (SDF) or application default definitions (ADF).

Step 6: Enable the Application for the SA z/OS Graphical Interface

If you want the new application to appear in any of the special views on the NMC workstation, you need to update the member in the DSIPARM data set that holds the BLDVIEWS cards for the sysplex that your application will run in.

If you want the application to appear in an existing view, you need to add a NONSNA statement:

```
NONSNA=plexname.subsysname/APL/sysname*,
QUERYFIELD=MYNAME
```

where *plexname* is the name of your sysplex, *subsysname* is the 11-character subsystem name of your application, and *sysname** is a wildcard for the system names that you want to see the application in this view.

If you want to add a new view, you will need to add a view statement:

```
VIEW=ING.plexname,
ANNOTATION='view description'
```

This needs to be followed by the NONSNA statement for the application as described above.

Step 7: Reload MPF List and Automation Configuration Files

Reload the MPF list and automation configuration files to enable automation of the application.

To reload the MPF list, type the following command:

- From the z/OS console:

  ```
  SET MPF=xx
  ```

- From a NetView console using the MVS prefix:

  ```
  MVS SET MPF=xx
  ```

where *xx* is the suffix of the MPF member in the SYS1.PARMLIB data set to load.

To reload the automation manager configuration file, all updated automation control files and the automation tables issue:

```
INGAMS REFRESH
```

and specify a data set name or an * which means reload the current one.

If SDF tree structures and panels have been loaded dynamically, you do not have to recycle SDF to have the application reflected in SDF at this point.

When you have completed these steps, the application is added to your automation policy and environment, and can be monitored using SDF.

Chapter 2. How to Create Automation Procedures

You can write additional automation procedures to supplement the basic automation procedures that are supplied by SA z/OS. For example, you may want to develop procedures to automate an application used exclusively on your system or to perform specialized automated operations for a subsystem.

SA z/OS generic routines and common routines perform basic functions such as logging messages and checking automation flags. You can use them in your own automation procedures.

SA z/OS generic routines and common routines are convenience routines that provide your automation procedures with a simple, standard way of interfacing with the automation control file, automation status file, and NetView log file. It is strongly recommended that you use these routines wherever possible in your own code.

"How CLIST or REXX Automation Procedures Are Structured" on page 6 describes how to structure your automation procedures. Refer to *IBM Tivoli System Automation for z/OS Programmer's Reference* for detailed descriptions and examples of the generic routines, common routines and file manager commands you can use in your automation procedures.

How Automation Procedures Are Called

There are several ways to call an automation procedure including:

- Calling the automation procedure from the NetView automation table using SA z/OS generic routines
- Keying in the automation procedure name or its synonym into a NetView command line
- Calling the automation procedure from another program
- Starting the automation procedure with a timer
- Starting the automation procedure with the NetView EXCMD command
- Starting the automation procedure on an automation operator with the SA z/OS AOFEXCMD command routine
- In the customization dialog, entering your automation procedure name in the **Command text** or **Command** field of the following entry types:
  - Application
  - MVS Component
  - Timers
  - Monitor Resources

Note: Not all routines can be called through all interfaces as some require extensive environmental setup before they are invoked.

How CLIST or REXX Automation Procedures Are Structured

It is recommended that the structure of automation procedures contain three main parts, as follows:

1. Perform initialization processing
2. Determine whether automation is allowed
3. Perform automation processing.

Figure 1 illustrates the structure of automation procedures for system operations and Figure 2 for processor operations.

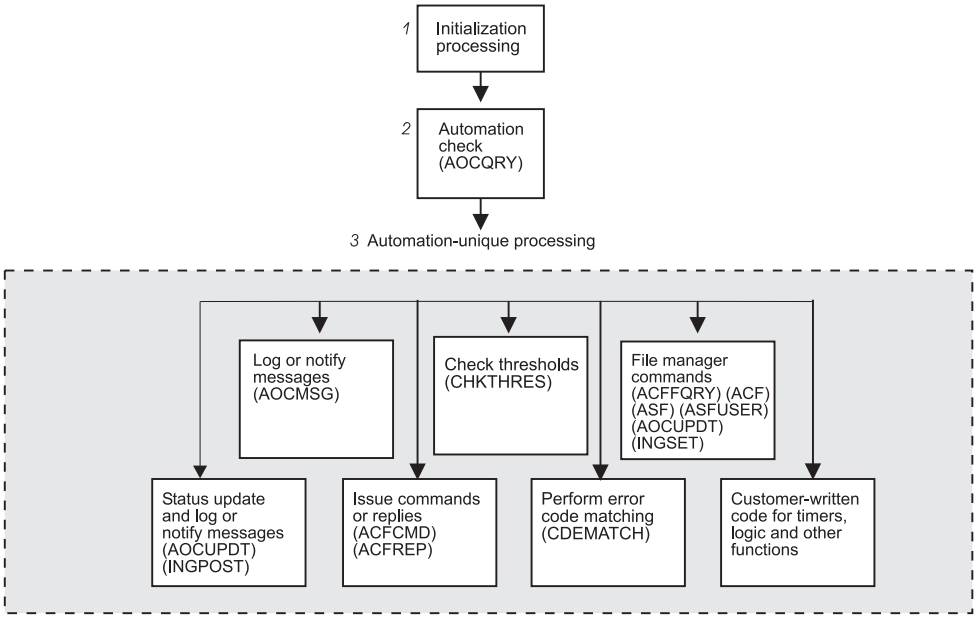


Figure 1. Automation Procedures for System Operations

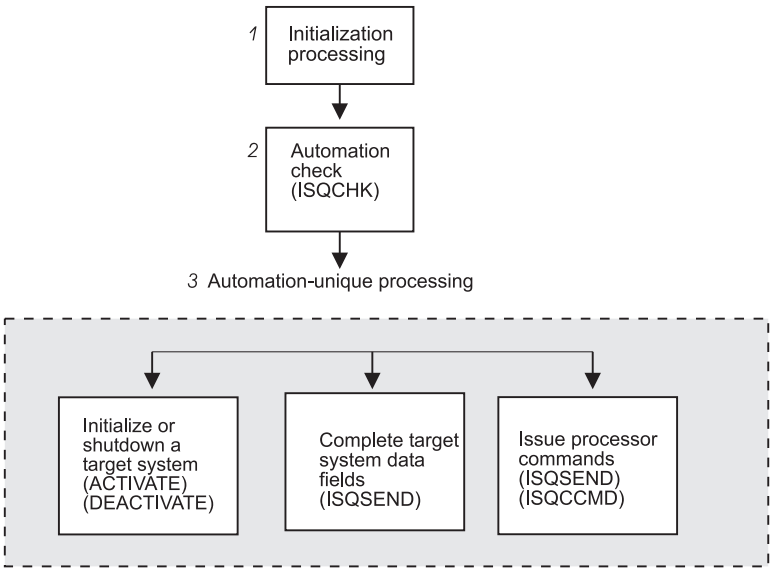


Figure 2. Automation Procedures for Processor Operations

The following sections provide more details about each part of an automation procedure.

Performing Initialization Processing

Initialization processing may not be required for simple automation procedures.

Initialization processing is responsible for:

- Setting up any error trap routines.
- Identifying the automation procedure by setting a local variable either explicitly or at execution time. This step makes it simpler to code routines that log messages and send notifications.

  From REXX™, the name of the CLIST is returned by the "parse source" statement.
- Declaring the global variables, such as common and task global variables, that are used for subsystem definition values in CLIST.

  See Appendix A, "Global Variables," on page 235 for descriptions of global variables. In REXX, you must GET the first time you reference the globals, and PUT when you update them.
- Checking to see if debugging (general or CLIST-specific) is on.
- Issuing debugging messages, if debugging is turned on.
- Validating the automation procedure call.

  This step can help prevent an operator from calling the automation procedure inappropriately. Automation procedures can also be validated using command authorization checking methods provided by NetView or an SAF product.
- Saving NetView message parameters. This step is necessary if your automation procedure uses the NetView WAIT statement and you need to access the original message text or control information.

For more information on coding automation procedure initialization sections, refer to "Example Automation Procedure" on page 14, to *Tivoli NetView for z/OS Customization Guide* and to *Tivoli NetView for z/OS Automation Guide*.

Determining whether Automation Is Allowed

System Operations

Automation procedures for applications and MVS components that are called from the NetView automation table should always perform an automation check by calling the AOCQRY common routine. AOCQRY checks that the automation flags allow automation. These checks eliminate the risk of automating messages for applications that should not be automated, or for which automation is turned off. AOCQRY also initializes most of the common and task global variables that are used in the automation procedure with values specific to the application.

Refer to *IBM Tivoli System Automation for z/OS Programmer's Reference* for more information on coding the automation check routine.

Processor Operations

Most of the processor operations commands run only when processor operations has been started. To determine whether processor operations is active, you can use the ISQCHK command in your automation routines. If processor operations is not running, ISQCHK returns return code 32 and issues the message:

```
ISQ0301 Cannot run cmd-name command until Processor Operations has started.
```

Your application can then issue the ISQSTART command to begin processor operations.

Performing Automation Processing

Automation processing is performed by any combination of SA z/OS routines and your own code. The following documentation gives more information on coding automation procedures:

- "Automation Processing in System Operations"
- "Automation Processing in Processor Operations" on page 9

Automation Processing in System Operations

This section contains information on how to customize automation processing for system operations.

Updating Status Information: You can update status information by calling the AOCUPDT common routine. This routine is used when a message indicates a status change. This would normally be done from the generic routines ACTIVMSG, HALTMSG, and TERMMSG. Making your own status updates may cause unpredictable results.

For more information, see *IBM Tivoli System Automation for z/OS Programmer's Reference*.

Logging Messages and Sending Notifications: You can log messages and send notifications by calling the AOCMSG common routine.

AOCMSG will:

- Format a message for display or logging
- Issue messages as SA z/OS notification messages to notification operators.

For more information, see *IBM Tivoli System Automation for z/OS Programmer's Reference*.

Issuing Commands and Replies: You can issue commands and replies by calling the ACFCMD and ACFREP common routines. You can use these routines to:

- Issue one or more commands in response to a message.
- Issue a single reply in response to a message.
- Use the step-by-step (PASS) concept to react to or recover from an automation event.

ACFCMD issues one or more commands. It supports both a single reaction and the step-by-step (PASS) concept. For more information, see *IBM Tivoli System Automation for z/OS Programmer's Reference*.

ACFREP issues a single reply. It supports both a single reaction and the step-by-step (PASS) concept. For more information see *IBM Tivoli System Automation for z/OS Programmer's Reference*.

In many cases you may be able to use the ISSUEACT generic routine that also supports single and pass processing.

Checking Thresholds: You can check and update thresholds by calling the CHKTHRES common routine. Use CHKTHRES to track and maintain a threshold,

and to change the recovery action based on the threshold level exceeded. For more information see *IBM Tivoli System Automation for z/OS Programmer's Reference*.

Checking Error Codes: You can check error codes by calling the CDEMATCH common routine. Use CDEMATCH to compare error codes in a message to a set of automation-unique error codes to determine the action to take. For more information, see *IBM Tivoli System Automation for z/OS Programmer's Reference*.

In some cases you may be able to use the code matching capabilities of the ISSUEACT and TERMMSG generic routines.

Using File Manager Commands: You can use file manager commands to access SA z/OS control files such as the automation control file and automation status file. Use ACF if you need to load or display the automation control file. Use ACFFQRY to query the automation control file quickly. Use ASF to display the automation status file. Use ASFUSER to modify the automation status file fields reserved for your own information. For more information, see *IBM Tivoli System Automation for z/OS Programmer's Reference*.

Using External Code for Timers, Logic, and Other Functions: Your automation procedures may require code to set timers, to perform logic unique to your enterprise or to the automation procedure itself, and to perform other functions. Some examples include:

- Issuing commands and trapping responses.

  You can issue commands and trap responses using the NetView WAIT or PIPE commands. You may need to use these commands in your code if it is necessary to check the value or status of a system component or application before continuing processing. For more information, see *Tivoli NetView for z/OS Customization Guide*.
- Setting Common Global and Task Global values to control processing.

  You can set Common and Task Global values by using NetView commands. You may need to set these values if it is necessary to set a flag indicating progress, message counts, and other indicators that must be kept from one occurrence of a message to the next. See *IBM Tivoli System Automation for z/OS Defining Automation Policy* for a table of all externalized SA z/OS global variables.

  Also refer to the discussion of common and task global variables in *Tivoli NetView for z/OS Customization Guide*.
- Setting timer delays to resume processing.

  You can set timer delays by using the NetView AT, AFTER, EVERY and CHRON commands. You can use these commands when an automation procedure must either resume processing or initiate another automation procedure after a given time to do additional processing. For example, you could use these commands to perform active monitoring of subsystems. For more information, see the discussion of AT, AFTER, EVERY and CHRON commands in *Tivoli NetView for z/OS Automated Operations Network User's Guide*.

Automation Processing in Processor Operations

This section contains information on how to customize automation processing for processor operations.

Initializing a Target System: If your routines need to start target systems (hardware and/or operating system), issue the ISQCCMD ACTIVATE command.

Shutting Down a Target System: If your routines need to shut down a target system, issue the ISQCCMD DEACTIVATE OCF command. Before issuing the

command to close the target system, shut down all of your functioning subsystems. This avoids any unexpected situations at the target system.

Issuing Other OCF Commands: All OCF commands supported by processor operations can be issued from automation routines. See *IBM Tivoli System Automation for z/OS Operator's Commands* for details about these commands.

Reserved SA z/OS Commands: The SA z/OS commands ISQISUP, ISQISTAT, ISQCMMT, ISQSTRT, ISQXIPM, ISQGPOLL, and ISQGSMSG are not intended for your use. Do not use these in your automation routines. Unexpected results may occur.

The following commands can only be used from an operator console and should not be used in your automation routines or with ISQEXEC: ISQXDST, ISQXOPT, and ISQHELP.

The following commands are for automation and should not be used in your automation routines: ISQI101, ISQI212, ISQMCLR, ISQI320, ISQIUNX, ISQI347, ISQI470, ISQI886, ISQI888, ISQI889, ISQI128, ISQIVMT, ISQMVMI1, ISQMVMI2, ISQMWAIT, ISQMDCCF, ISQM020, and ISQIPLC.

Serializing Command Processing: Serializing command processing ensures that commands and automation routines are processed in the order in which they are sent to a target system console. It can also prevent the command sequence from being interrupted by other tasks.

Specific target control tasks are assigned to specific target systems during initialization of the target system. More than one target system can share a target control task, but a target system never has more than one target control task allocated to it to perform work.

When a command or an automation routine is sent to a target system, it can be processed partly in the issuing task (a logged-on operator or an autotask) and partially in a target control task. When the command or automation routine is to be processed by a target control task, it is either allocated to the target control task and processed, or queued to be processed by the target control task. This serializes the processing of commands and automation routines. Serializing ensures that they are processed in the order in which they were sent to the target system console.

The NetView program has priority defaults established during its initialization. Usually, everything running under NetView has a low priority. You can use the NetView DEFAULTS command to see what the settings are, but you should not change them. For SA z/OS command processing to be serialized as designed, all commands used in SA z/OS must have a priority setting of "low". If you change the priorities or have more than one priority for commands used in SA z/OS, the difference in the priorities may defeat the serialization that results from the architecture of the target control task.

Sending an Automation Routine to a Target Control Task: If you run the same series of SA z/OS commands regularly, you can program the commands into a NetView automation routine. Follow the guidelines you use for any NetView automation routine.

A NetView autotask or a logged-on operator can then run this routine or send it to a target control task. Use the following command to transfer an automation routine to a target control task:

```
ISQEXEC target-system-name SC routine-name
```

When you issue the ISQEXEC command to process an automation procedure, all of the commands are processed in the order in which they occur in the automation procedure. This is because the ISQEXEC command sends work to a target control task, which processes commands serially. Any other commands or automation routines issued to the same console by the ISQEXEC command are queued for processing by the target control task and do not start until the previous command or automation procedure completes.

The ISQEXEC command also frees the original task from any long-running command sequence. This lets you use the issuing task, such as an OST, for other work.

The ISQEXEC command does not lock consoles to ensure command serialization; the command serialization process is due to the target control task allocation scheme. Commands and automation routines are processed in the order in which they occur; however, it is possible for commands from other tasks to interrupt the command sequence.

For more information about the ISQEXEC command, see *IBM Tivoli System Automation for z/OS Operator's Commands*.

Locking a Console: Several routines and operators may attempt to address the same console at the same time. The ISQEXEC command does not prevent other tasks from interrupting the sequence of commands being processed by the target control task; it does not lock the console.

To prevent a sequence of commands from being interrupted, use the ISQXLOC and ISQXUNL commands. The ISQXLOC command locks access to the console. If a task attempts to issue a command to a locked console, the task is told that the console is locked, and the command fails. When you are finished with the sequence of commands that must be processed without interruption, issue the ISQXUNL command to unlock access to the console.

You can use the ISQXLOC and ISQXUNL commands within automation routines to ensure that they complete without interference from other tasks. For automation routines that issue a number of SA z/OS commands, put the following command after the ISQEXEC command and near the beginning of the routine:

```
ISQXLOC target-system-name SC
```

This locks access to the target system console to the current task until the lock is dropped by the command:

```
ISQXUNL target-system-name SC
```

Only the task that issued ISQXLOC can successfully issue ISQXUNL. If an ISQXLOC command is issued from a locked sequence of commands, it is rejected because the console is already locked.

When you lock a system console for a target system running on a logical partition, you lock that system console for all other target systems using that processor. A command sent to a system console for any other target system (logical partition) on that target hardware definition will not run until the console is unlocked.

If your automation routine cannot wait for a console to be released, use the ISQOVRD command to gain control of the console. Use the following command only in **critical** automation routines:

```
ISQOVRD target-system-name SC
```

When the routine issuing the override command completes, the lock is removed and the console is available.

How to Make Your Automation Procedures Generic

By using the SA z/OS common routines, you can make your own automation procedures generic. A generic automation procedure comprises three parts. For each part, there are special common routines that help you to fulfill your tasks:

Preparation
Check if automation is allowed and should be done. Use common routine AOCQRY.

Evaluation
What should be done? Use common routine CDEMATCH.

Execution
Do what should be done. Use common routines ACFCMD or ACFREP.

```
*****************************************
*******          Preparation         *******
*****************************************

AOCQRY

- check if the resource is controlled by SA z/OS

- check if automation is allowed

- prepare/set task global variables for CDEMATCH, ACFCMD and ACFREP

...

CDEMATCH

- code matching (table search in ACF)

- find out required action

...

ACFCMD/ACFREP

- do required action:
  issue command / respond reply
```

Figure 3. Skeleton of an Automation Procedure

For more information on the mentioned common routines refer to *IBM Tivoli System Automation for z/OS Programmer's Reference*. For more information on command processing or reply processing refer to *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Processor Operations Commands

Whenever possible, your automation routines should make use of SA z/OS's processor operations OCF commands, also called common commands. These

commands are independent of the hardware type of the target system's processor. Therefore, the use of these commands minimizes the need for changes to your automation routines if you need to add new processors to your configuration. See *IBM Tivoli System Automation for z/OS Operator's Commands* for a detailed description of the processor operations commands.

Developing Messages for Your Automation Procedures

Depending on the scope of additional programming, creating new automation procedures may also require developing additional messages.

Some SA z/OS facilities and commands you can use to develop messages include:

- The AOCMSG common routine (see *IBM Tivoli System Automation for z/OS Programmer's Reference*).
- The AOCUPDT common routine (see *IBM Tivoli System Automation for z/OS Programmer's Reference*).

The following steps summarize the message development process.

1. Choose a message ID. Make sure it is unique.
2. Use NetView message services to define the message to NetView.

   Put an entry for the message in a DSIMSG data set. This data set must be identified in a DSIMSG data definition (DD) name.
3. Use the AOCMSG common routine to issue the message (see *IBM Tivoli System Automation for z/OS Programmer's Reference*).
4. Add an entry for the message to your production copy of the NetView DSIMSG data set.

Example AOCMSG Call

This example shows how to code AOCMSG to issue message ABC123I.

Entries for messages in DSIMSG member DSIABC12 are as follows:

```
*******************************************************
120I ...
121I ...
122I ...
123I 10 40 THE EAGLE HAS &1
124I ...
*******************************************************
```

Your automation procedure contains the following AOCMSG call:

```
<other automation procedure code>
.
.
.
     AOCMSG LANDED,ABC123
.
.
.
<other automation procedure code>
```

When AOCMSG is called as specified in the automation procedure, DSIMSG member DSIABC12 is searched for message ABC123I. Substitution for variable &1 occurs, and the following message is generated:

```
ABC123I THE EAGLE HAS LANDED
```

Note that the message is defined with a 10 and a 40 between the message ID and the first word of the message. These are the SA z/OS message classes to which the message belongs. When the message is issued a copy is sent to every notification operator who is assigned class 10 or class 40 messages.

Refer to *Tivoli NetView for z/OS Customization Guide* for further information on developing new messages.

Example Automation Procedure

This section provides an example of an application program that handles a z/OS message. The automation procedure uses a subset of the SA z/OS common routines or generic routines.

```
/* Example SA z/OS Automation Procedure */

1  Signal on Halt Name Aof_Error; Signal on Failure Name Aof_Error
   Signal on Novalue Name Aof_Error; Signal on Syntax Name Aof_Error

2  Parse source .. ident .

3  "GLOBALV GETC AOFDEBUG AOF."||ident||".0DEBUG AOF."||ident||".0TRACE"
   If AOFDEBUG = 'Y' Then
     "AOCMSG "||ident||",700,LOG,"||time()||","||opid()||","||Arg(1)
   loc.0debug = AOF.ident.0DEBUG
   loc.0trace = AOF.ident.0TRACE
   loc.0me    = ident
   If loc.0trace <> '' Then Do
     loc.0debug = ''
     Trace Value loc.0trace
   End

4  save_msg  = msgid()
   save_text = msgstr()
   lrc = 0

5  /* This procedure can only be called for msg IEA099A */
   If save_msg <> 'IEA099A' Then Do
     "AOCMSG "||loc.0me||",203,"||time()||","||opid()
     Exit
   End

6  "GLOBALV GETC AOFSYSTEM"
   cmd = 'AOCQRY '||save_msg||' RECOVERY '||AOFSYSTEM
   cmd
   svretcode = rc
   If loc.0debug = 'Y' Then
     "PIPE LIT /Called AOCQRY; Return Code was "||svretcode||"/" ,
       "| LOGTO NETLOG"

   /* ------------------------------------------------------------- **
   ** Check return code from AOCQRY                                 **
   ** 0 = ok                      1 = global flag off               **
   ** 2 = specific flag off       3 = resource not in ACF           **
   ** 4 = bad parms               5 = errors/timeout                **
   ** ------------------------------------------------------------- */
   Select
7    When svretcode >= 3 Then Do
        "AOCMSG "loc.0me",206,,"time()",,,"cmd",RETCODE="svretcode
        lrc = 1
     End
8    When svretcode > 0 Then Do
       "GLOBALV GETT AUTOTYPE SUBSAPPL SUBSTYPE SUBSJOB"
       "AOCMSG "loc.0me",580,,"time()","SUBSAPPL","SUBSTYPE"," ,
         SUBSJOB","AUTOTYPE","save_msg
       lrc = 1
     End
     Otherwise Do
9      Parse Var save_text With . 'JOBNAME=' save_job 'ASID=' save_asid .

10     ehkvar1 = save_job
```

```
        ehkvar2 = save_asid
        "GLOBALV PUTT EHKVAR1 EHKVAR2"
11      cmd = 'ACFCMD ENTRY='||AOFSYSTEM||',MSGTYP='||save_msg
        cmd
        svretcode = rc
        If loc.0debug = 'Y' Then
          "PIPE LIT /Called ACFCMD; Return Code was "||svretcode||"/" ,
            "| LOGTO NETLOG"

        /* ---------------------------------------------------------- **
        ** Check return code from ACFCMD                              **
        ** 0 = ok                        1 = no commands found in ACF  **
        ** 4 = bad parms                 5 = errors/timeout            **
        ** ---------------------------------------------------------- */
12      If svretcode > 1 Then Do
          "AOCMSG "loc.0me",206,,"time()",,,'"cmd"',RETCODE="svretcode
          lrc = 1
        End
      End
    End /* End of Select svretcode */

13  Exit lrc

14  Aof_Error:
     Signal Off Halt; Signal Off Failure
     Signal Off Novalue; Signal Off Syntax
     errtype = condition('C')
     errdesc = condition('D')
     Select
       When errtype = 'NOVALUE' Then rc = 'N/A'
       When errtype = 'SYNTAX' Then errdesc = errortext(rc)
       Otherwise Nop
     End
     "AOCMSG "errtype",760,,"loc.0me","sigl","rc","errdesc
    Exit -5
```

Notes on the Automation Procedure Example

1 This step sets error traps for negative return codes, operator halt commands, and REXX programming errors.

2 This step defines the identity of the automation procedure.

3 This step handles the debug and trace settings (refer to “Using AOCTRACE to Trace Automation Procedure Processing” on page 17.

4 Save the NetView message variables the automation procedure uses.

5 Perform authorization check. This procedure can only be called for a particular message.

6 This section performs the automation check:

1. Fetch the AOFSYSTEM CGlobal that contains the information under which entry name the system messages are stored in the automation control file (ACF).
2. The automation procedure calls the AOCQRY common routine. This routine performs the automation flag check and presets some task global variables that are used by other common routines like ACFCMD.

7 Issue message AOF206I if call to AOCQRY fails.

8 Issue message AOF580I if automation flag is off.

9 Get the job name and asid reported in the message.

10 Set EHKVARn variables for ACFCMD.

11 Call ACFCMD to issue the command specified in the ACF. The Automation Control File entry for the message IEA099A could look like this:

```
MVSESA IEA099A,
CMD=(,,'MVS C &EHKVAR1,A=&EHKVAR2')
```

12 Issue message AOF206I if call to ACFCMD fails.

13 Exit with return code that indicates successful or unsuccessful processing.

14 This code logs a message if an error is trapped at step **1** .

Installing Your Automation Procedures

The installation process for a new automation procedure depends on the language in which the automation procedure is written.

- If the automation procedure uses a compiled language, such as PL/I, C, or Assembler:
  1. Compile or assemble your source into an object module.
  2. Link-edit the object module into a NetView load library.
  3. Include an entry for the automation procedure in the DSICMD member of the NetView DSIPARM data set.
- If the automation procedure uses an interpreted language such as NetView command list or REXX:
  1. Copy the automation procedure into a NetView command list library
  2. Optionally include an entry for this automation procedure in the DSICMD member of the NetView DSIPARM data set. Then it is more quickly found and invoked.

For more information on preparing your code for use and installing it, refer to *Tivoli NetView for z/OS Customization Guide*

Testing and Debugging Automation Procedures

This section describes SA z/OS and NetView facilities you can use for testing automation procedures, including:

- SA z/OS assist mode
- SA z/OS AOCTRACE operator facility
- NetView testing and debugging facilities

The Assist Mode Facility

SA z/OS provides an *assist mode* facility, so that you can verify actions of automation procedures and automation policy before letting them run in a completely automated environment.

When assist mode is on, actions that are normally taken by SA z/OS automation procedures, such as issuing a command or reply or calling a common routine, are not performed. Instead messages that describe what would have happened are written to the netlog.

The assist mode is associated with automation flags (Automation, Initstart, Start, Recovery, Terminate or Restart). Whether assist mode is used for any action is determined by the automation flag. This is checked to see whether that action is permitted.

Cases where you might want to use assist mode include:

- During early stages of developing and using your automation policy
- After changing your automation policy, such as after adding an application to automation
- After adding a new automation procedure to the SA z/OS code

Using Assist Mode to Test Automation Procedures

Assist mode can help you to detect problems with your automation procedures before they are added to your production code. Assist mode works by intercepting commands and replies before they are issued through NetView. The intercepted commands and replies, as coded in the automation policy, are reformatted into a message that is sent to the NetView log.

The reformatted command is issued in message AOF320I and the reformatted reply in message AOF323I. Each message contains detailed information about the action defined in the automation policy and the actual action to be issued.

During run time of SA z/OS, the assist mode can be enabled with the INGAUTO command to set the related automation flag to the value L. The DISPFLGS command can be used to view the current automation flag settings. Any other value for the automation flag deactivates assist mode.

When an event triggers an automated action and assist mode is enabled, SA z/OS logs the action in the NetView log. The log can be reviewed to ensure that automation has run as expected.

Assist mode works for all routines that call the SA z/OS common routine, after having checked the automation flag by calling AOCQRY.

Using AOCTRACE to Trace Automation Procedure Processing

The AOCTRACE command dialog maintains both global execution flow traces and automation procedure (CLIST) specific debugging flags. Setting the global flag causes all routines that support tracing and all message IDs to record a statement in the NetView log whenever they are invoked. The AOFDEBUG global variable is used to pass the global flag information to the CLIST. The global flag is set to null if the global trace is off, or Y if the global trace is on.

Setting the CLIST-specific flags lets you obtain information about what the CLIST is doing when it executes, or lets you activate a REXX trace. The debug flag is either null or Y, and is stored in the AOF.*clist*.0DEBUG common variable (where *clist* is the true CLIST name).

The trace flag is set to null or a valid REXX trace type, as follows:

- A (All)
- R (Results)
- I (Intermediate)
- C (Commands)
- E (Errors)
- F (Failures)
- L (Labels)
- O (Off)
- N (Normal)

The S (Scan) trace type cannot be used.

The trace flag is stored in the common global variable AOF. *clist*.0TRACE (where *clist* is the true CLIST name).

Message tracing can only be set from the command line, using the command `AOCTRACE MSG/id,ON|OFF` where *id* is the message to be traced.

AOCTRACE is documented in *IBM Tivoli System Automation for z/OS Operator's Commands*.

REXX Coding Example

For examples of code that can be placed at the beginning and end of your REXX automation routines to handle trace and debug settings, see AOFEXC00 in the SINGSAMP library.

When writing code to support the debug feature, you should expose *loc.* on all your procedures and insert fragments of code to check the value of the *loc.0debug* flag and output relevant information. The *loc.0me* assignment makes the CLIST name available everywhere, so you can prefix all debug messages with it. You can then tell where the messages are coming from. For example:

```
Myproc:
  Procedure expose loc.
  If loc.0debug = 'Y' Then
     'PIPE LIT /' loc.0debug ' has called procedure MYPROC/',
     '| LOGTO NETLOG'
  Return
```

NetView Testing and Debugging Facilities

NetView provides several facilities to assist in testing and debugging automation procedures.

To do detailed testing, you may want to trace every statement issued from automation procedures. This type of testing is enabled through the &CONTROL statement for NetView command lists and through the TRACE statement for REXX procedures.

You can also specify less detailed tracing on the TRACE and &CONTROL statements, so that only commands are traced. A comparable facility, the interactive debugging aid, is available for programs coded in PL/I and C.

Perform specific tracing by issuing NetView MSG LOG, PIPE LOGTO NETLOG commands at appropriate points throughout a NetView command list, REXX procedure, or PL/I routine.

To test for proper parsing and reaction to a message, write a short automation procedure to issue a NetView WTO command. This WTO is processed by the NetView automation table and triggers the appropriate automation procedure. If the automation procedure requires the job name, the job name must be temporarily hard-coded to the appropriate name. In this case, because the WTO was issued from the NetView region, the job name associated with the message is the NetView region. A sample automation procedure follows:

```
WRITEWTO CLIST
   WTO &PARMSTR
   &EXIT
```

The sample automation procedure can issue any single-line message by calling the routine. For example, to issue message ABC123I, which indicates the start of a program, the command is:

```
WRITEWTO ABC123I My testprogram PRGTEST has started.
```

Where to Find More Testing Information

More information on testing can be found in the following books:

- *Tivoli NetView for z/OS Customization Guide*

  This book lists requirements for your programs, including preparing your code for use, and detailed information on writing exit routines and command processors.
- *Tivoli NetView for z/OS Automation Guide*

  This book has guidelines for creating new automation procedures, including a recommended development process.

Coding Your Own Information in the Automation Status File

You can code your own information in the automation status file with the ASFUSER command.

The automation status file has 40 user data fields that are associated with each resource that is defined within it. You may use these fields to store persistent information about resources that your code needs to access later. The information in the ASF is not lost when SA z/OS is shut down. It will last until either of the following occurs:

- The ASF VSAM data set is deleted and redefined,
- You bring SA z/OS up with an automation control file that does not include the application that the information has been defined for

Note that you should verify that the information you have stored in the automation status file is accurate whenever SA z/OS initializes, as circumstances may have changed while SA z/OS was down.

Each automation status file field reserved for your data can contain up to 20 characters. The ASFUSER command allows you to update and display data in these fields. See *IBM Tivoli System Automation for z/OS Programmer's Reference* for the ASFUSER command description.

Programming Recommendations

This section contains tips and techniques that may help to reduce the coding effort required when writing your own automation procedures, and to improve performance of your automation procedures.

- Use variables, such as &IDENT, &SUBSAPPL, &SUBSTYPE, and &SUBSJOB in place of parameter values.

  Using &IDENT for automation procedure names allows for changes to automation procedure names (only the &IDENT variable value needs changing). The &SUBSxxx variables allow for subsystem and job name changes (changes to subsystem and job names need only be made in automation policy).

  Using NetView command list language variable JOBNAME for the resource field on an AOCQRY call, an automation procedure can be written to support a known message for any job that can issue a message.
- Use defaults when possible to minimize coding.

- Use generic error codes (see CDEMATCH).
- Use available message parsing techniques:
  - Use the NetView command PARSEL2R or REXX PARSE command to parse a message without relying on a field position in a message.
  - Parse a message in the NetView automation table and send only necessary fields to an automation procedure.
- Consider not coding the ENTRY field in CDEMATCH calls (default is the SUBSAPPL returned from the last AOCQRY call).
- Use appropriate automation flags.
- Review the coding requirements in *Tivoli NetView for z/OS Customization Guide* including restrictions to consider when writing code, such as:
  - Restrictions when TVBINXIT is on
  - Variable names
  - Macro use
  - Register use
  - Re-entering programs
- Use SA z/OS generic routines where possible, because they:
  1. Reduce your maintenance overhead.
  2. Often use internal interfaces that are more efficient than the common routines. Similarly, it is better to use a common routine than to write your own code to process the response from an ACF display request.
- Use SA z/OS's processor operations common commands where possible, because these:
  1. Are independent of the hardware type of the target system's processor
  2. Minimize the need for changes to your automation routines as you add new processors to your enterprise
- Consider using the NetView VIEW command to display online help text associated with new code, and to develop a fullscreen interface for new commands that are a part of the new code. Refer to *Tivoli NetView for z/OS Customization Guide* for information on the VIEW command.

Global Variable Names

When creating your own automation procedures, you must ensure that the names of any global variables you create do not clash with SA z/OS external or internal global variable names. SA z/OS external global variables are documented in *IBM Tivoli System Automation for z/OS Defining Automation Policy*. In addition, you should not use names beginning with:

- CFG
- AOF
- ING
- ISQ
- EVI
- EVE
- EVJ

Chapter 3. How to Add a Message to Automation

SA z/OS exploits the NetView automation table (AT) technique. The ATs contain traps for messages that must be automated. If an action must be taken in response to a message, this action needs to be defined in the customization dialog. A related AT entry is required to call a routine to execute the action.

SA z/OS automatically generates the ATs.

Conceptual Overview

This section gives a brief overview of the main aspects of SA z/OS message automation:

- A list of messages that are involved in SA z/OS automation is generated by SA z/OS. This can then be used as an MPF member.
- Message automation is a NetView AT-based process.
- ATs are generated by SA z/OS.
- AT entries will be created for messages where actions are defined for.
- Messages can be defined to indicate a status change.
- Messages can be marked to be ignored or suppressed, thus not generating an AT entry.
- Messages can be marked to be captured for further display
- Most AT entries trap messages independent of the issuing product instance, component or module.
- Predefined AT entries can be changed.
- You can define the AT scope to determine precisely if and what kind of ATs are built.
- The following action codes are available in message processing:

  CMD (C)
  Allows you to enter a command in response to a message.

  REP (R)
  Allows you to enter a reply in response to a message.

  COD (K)
  Allows you to enter codes that can be checked within a message to prompt a certain command.

  USR (U)
  Allows you to enter any user data in keyword-data pairs.

  AUT (A)
  Allows you to enter a resource status indicated by a message.

  OVR (O)
  Allows you to override a default AT entry that is generated for a message.

Defining Actions for Messages

AT entries are generated by SA z/OS for messages that are defined for APL, MTR or MVC policy entries and that have actions (for example, CMD or REPLY) defined.

Note: Throughout this chapter, whenever the term *policy entry* is used, it implies either an APL, MTR or MVC policy entry, unless otherwise stated.

There are two kinds of messages that influence the build of AT entries:

- ***Known*** messages — These are messages where SA z/OS provides specific automation that is unique for the given message (for example, IAT3011). Thus this message is *known* to SA z/OS). A single AT entry is predefined just for this known message.
- ***Unknown*** messages — These are messages where SA z/OS provides automation that is generic for messages that are *unknown* to SA z/OS. SA z/OS maintains wildcard message automation for those messages not having a specific automation defined. (For example, message IAT9999 is unknown to SA z/OS.) A wildcard niche within an AT is the place where unknown messages are placed.

The first step in defining actions is to select a policy entry from the Policy Selection panel. From its policy selection list, select the MESSAGES/USER DATA policy item. This leads to the Message Processing panel, where you can then define actions for message IDs. If an AT entry is built according to the action, it will only check for the message ID by default, independently of the product instance, component or module issuing that message. If this is not intended, you can use the OVR action (see "Defining OVR Actions" on page 24).

There are many messages that are known to SA z/OS. For these messages specific AT entries are predefined by SA z/OS. Here, the action defined in the customization dialog does not determine the AT entry. If you want to know what kind of AT entry is built for automating a particular message, you can check the generated AT fragment member after generating the AT.

Notes:

1. SA z/OS symbols (AOCCLONEs) and System Symbols should not be used for or within message IDs. Otherwise the correct sequence of entries within a generated AT cannot be guaranteed.
2. If an AUTO, REPLY or CMD action is defined for a message for an APL class and the same message ID is used to define a OVR action for an APL instance that is linked to that class, two AT entries are built for the message ID. The AT entry for the OVR action is built before the entry for the AUTO, REPLY or CMD action.

Defining CMD or REP Actions

Define a CMD or REP action for message XYZ222I in the CMD Processing or Reply Processing panel, where XYZ222I is a message that is unknown to SA z/OS.

This definition leads to the creation of an AT entry for message XYZ222I using the generic routine ISSUEACT after the next Configuration Build process.

Note: If you have code definitions that you expect to be passed to ISSUEACT, you have to manage the AT overrides to do this. This is *not* done by SA z/OS. See "Defining OVR Actions" on page 24.

Note that for MVC entries, unknown messages will have the parameter SYSTEMMSG=YES added to the SA z/OS generic routine (ISSUEACT). If the same message ID is defined for MVC and APL, the APL entry will cause the AT entry to be generated. No additional AT entry is built for the message ID that is defined for MVC.

Defining AUTO Actions

Defining Status Messages

Many messages that indicate a state change of APL, MTR, and MVC resources are known to SA z/OS. The related AT entries are already predefined. For these messages there is no need to define them in the policy database.

If necessary, you can define additional application messages that indicate a state change. You must do this for non-IBM or user application messages that indicate a state change. The AUTO action therefore leads to a selection panel that lists resource states.

The Status Message Report shows all Status Messages. It lists all user-defined and predefined Status Messages and their associated statuses.

Status messages can be defined for MVC policy entries as well as for APL and MTR instances or classes. The following description is for an UP status message based on an APL resource definition.

As an example, define an UP state indicated by message XYZ444I in the Message Type Selection panel. Here, XYZ444I is a message that is unknown to SA z/OS.

This definition leads to the creation of an AT entry for message XYZ444I using the generic routine ACTIVMSG after the next Configuration Build process.

Notes:

1. There are certain messages that can be used as Status Messages, but for some messages, CODE definitions are required (for example, IEF450I, HASP095, etc.). TERMMSG will set the status depending on these definitions. For more details about TERMMSG, see *IBM Tivoli System Automation for z/OS Programmer's Reference*.
2. If an AUTO, REPLY or CMD action is defined for a message for an APL class and the same message ID is used to define a OVR action for an APL instance that is linked to that class, two AT entries are built for the message ID. The AT entry for the OVR action is built ahead of the one for the other action.
3. Automation table entries are generated based on the messages that are defined with MESSAGES/USER data. For size and performance reasons, these entries are message-oriented rather than job-oriented.

   This means that an AUTO action (except IGNORE or SUPPRESS) for a particular message generates an AT entry. This entry traps that message independently of the issuing subsystem. It then sets the subsystem state as selected via the AUTO action.

   If a state message should be processed for a particular subsystem only, you can use an OVR action.

Defining Captured Messages

If messages only need to be captured to be displayed but not automated, the AUTO selection panel provides an additional CAPTURE function.

Messages that have a CMD or REPLY action defined for them or that are defined as Status Message are implicitly captured. There is no need to explicitly define these messages to be captured.

Define message XYZ555I to be captured in the Message Type Selection panel. Here XYZ555I is a message that is unknown to SA z/OS.

This definition leads to the creation of an AT entry for message XYZ555I using the generic routine AOFCPMSG after the next Configuration Build process.

Note: The status (AUTO) action is mutually exclusive with the OVR action.

Preventing the Building of AT Entries

Inhibiting AT and MPFLSTSA Entries: Using the AUTO action you can select IGNORE or SUPPRESS for certain messages:

- Messages that are marked IGNORE will not cause an AT entry or an MPFLSTSA entry to be generated.
- Messages that are marked SUPPRESS will not cause an AT entry to be generated. An MPFLSTSA entry is generated with the options SUP(YES),AUTO(NO).

IGNORE and SUPPRESS overrule other actions (except OVR) that are defined for the same message, even though these actions are defined on other PDB entries.

The MPFLSTSA member is built for each PDB. Because IGNORE and SUPPRESS affect the build of the MPFLSTSA member, these definitions also have a PDB-wide scope.

For example, if the following definitions are made within the same PDB then *no* MPFLSTSA entry is generated for ABC111I even though this entry is required for SYSPLEX1 or SYS1:

- The AT scope is set to SYSPLEX or SYSTEM
- A CMD is defined for message ABC111I on APL1 that is linked to SYS1 within SYSPLEX1
- IGNORE is defined for message ABC111I on APL2 that is linked to SYS2 within SYSPLEX2

AT Entries That Are Never Built: There are many keywords that can be entered as message IDs in the Customization Dialog (for example, message MVSDUMPFULL). No AT entry is built for these keywords. A list of these keywords is given in the online help.

Defining OVR Actions

You can apply an OVR action in the Message Processing panel to a message ID for an APL instance, APL class or an MVC PDB entry.

The OVR action allows you to preview an AT entry as it would be built according to the actions that are defined for a message of an APL or MVC policy entry.

If you are using the OVR action to preview an AT entry for a message that is unknown to SA z/OS and where no other action (CMD, REPLY or AUTO) is defined, no AT entry is predefined. The condition and action fields of the Automation Processing panel are empty.

The OVR action allows you to override an AT entry. The condition and action statements of an AT entry can be changed. Action statements can be added or deleted. Deleting the condition statement will remove the AT override.

AT entries cannot be changed by an OVR action if an AT entry is *forced* by SA z/OS or if there is already an AUTO action defined for that message on the same policy entry.

You can define '&SUBSJOB' as part of an AT condition statement that will be replaced by the job name of the given policy entry when building the AT. This is very valuable when defining an AT entry for an APL class. Then each APL instance linked to that class will have its own AT entry with its job name in the AT condition statement. Checking for the job name may also be required if different instances of a product issue the same message but you want only certain jobs to be affected by that message.

SA z/OS symbols (AOCCLONEs) and system symbols may be contained in an AT override definition. They will be resolved at AT load time.

Defining an OVR action for message XYZ666I (that is unknown to SA z/OS) in the Message Processing panel leads to the Automation Processing panel. Here you can either change a predefined AT entry that then becomes a user-defined AT entry, or, if no predefinitions are available, you can define a user specific AT entry. If message XYZ666I should be trapped, enter `MSGID = 'XYZ666I'` in the NetView AT condition field. If routine MYREXX1 should be called in that case, enter for example:

```
EXEC(CMD('MYREXX1')ROUTE(ONE %AOFOPWTORS%))
```

This definition leads to the creation of an AT entry for message XYZ666I using the routine MYREXX1 after the next Configuration Build process.

Note: The location of the AT entry within the NetView automation table is determined by the Message ID that is defined in the Message Processing panel and not by the **NetView AT condition** field in the Automation Processing panel.

Defining the NetView AT Scope

In the Edit Policy Data Base Entry panel in the customization dialog, the entry field AT Scope allows you to define the scope of a NetView Automation Table. Valid AT scope values are:

NONE
No AT or MPFLSTSA member will be built at configuration build time. Use this value if you want to maintain ATs yourself.

ENTERPRISE
One AT will be built to be shared within the whole enterprise.

SYSPLEX
One AT will be built to be shared within a sysplex.

SYSTEM
One AT will be built for each system of the selected PDB. (This is the default.)

If the AT scope changes from NONE to SYSTEM, a build of type ALL is required.

If the AT Scope is set to SYSPLEX, a standalone system must be linked to a sysplex group otherwise no AT is built for that system.

The AT Scope for MPFLSTSA is always ENTERPRISE or NONE.

Build

Once you have made all the message definitions you need, you can start the Configuration Build Process to build the configuration files containing the NetView Automation Table. For more information about the build function, refer to *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

When building the NetView ATs for the first time, the Build Type field in the Build Options section of the Build Parameters panel must be set to ALL, for example:

```
Build options:
  Output Data Set . . . . 'OPER.OUTPUT.CONFIG'
  Mode  . . . . . . . . . ONLINE      (ONLINE BATCH)
  Type  . . . . . . . . . ALL         (MODIFIED ALL)
  Configuration . . . . . NORMAL      (NORMAL ALTERNATE)
```

The AT fragments and the MPFLSTSA member will be built into the configuration data output data set.

This may require more space than you have allocated for the output data set. Thus enlarging the output data set may be required.

This also applies to the DSILIST data set where the AT listings are stored.

It is recommended that you copy the build output to a Generation Data Group (GDG) to avoid token mismatch conditions and AT load errors.

NetView Automation Table Build Concept

This section covers the following:

- When is an AT built? (See "When Is an AT Built?.")
- Predefined message automation (see "Predefined Message Automation" on page 27)
- The AT entry sequence (see "AT Entry Sequence" on page 29)

When Is an AT Built?

An AT is built depending on the following conditions:

- If the AT Scope has *not* been set to NONE. (If it has been set to NONE then neither an AT nor an MPF list is built.)
- Depending on the Build Type, ATs will either always be built or only be built in the case of a policy modification:

  ALL All ATs will be built.

  MOD Only ATs are built where changes have been made in the PDB that affect those ATs.
- For any Build Option:
  1. Build a complete enterprise
  2. Build sysplex group or standalone system
  3. Build entry type or entry name (if changes have been made to the AT, a complete enterprise build is made)

The following changes may affect the AT, thus causing an AT rebuild:

- Defining the first CMD, REPLY, CODE, USER, AUTO, or OVR action for a message ID, or deleting the last action
- Changing or deleting a message ID that has at least one of the above actions defined for it
- Changing AUTO or OVR definitions
- Changing the link between an APL, MVC, or MTR and a system
- Changing the link between an APL instance and a class
- Installing a new version of the internal AT build template (when applying service)
- Changing the job name
- Changing the AT Scope to SYSTEM, SYSPLEX or ENTERPRISE

These changes only affect the AT build if the APL, MVC, or MTR entries are linked to a system.

Note: If the MPF Header of Footer definitions have changed, an automatic AT build is not performed.

Predefined Message Automation

SA z/OS provides predefined message automation for messages that are known to SA z/OS.

The type of AT entry defines *whether* an AT entry is built for a particular message. There are three AT entry types:

- *Forced* AT entries—Always builds an AT entry. Modifications are not allowed. See "FORCED AT Entry Type" on page 263.
- *Recommended* AT entries—Always builds an AT entry. Modifications are allowed. See "RECOMMENDED AT Entry Type" on page 263.
- *Conditional* AT entries—Only builds an AT entry if the message is defined in the Customization Dialog.

  CODE and USER actions generate AT entries only for those messages that are known to SA z/OS. See "CONDITIONAL AT Entry Type" on page 264.

There are also other specialized AT entries (for example, for message IEF403I) that:

- Are always built because they are critical to the structure of the AT, see "AT Entries Built for Messages Known to SA z/OS" and "AT Entries for SA z/OS Internal Messages" on page 28
- Are never built because they contain SA z/OS keywords, see "AT Entries That Are Never Built" on page 24
- Are related to specialized activities, see "AT Entry Specialties" on page 28
- Have multiple actions defined per policy database entry, see "AT Entries for Messages That Have Multiple Actions Defined" on page 28

AT Entries Built for Messages Known to SA z/OS

A messages that is known to SA z/OS will always cause an AT entry to be generated if it is defined as a forced entry. These entries are critical to SA z/OS for proper functioning. In certain cases CMD and REPLY actions are allowed and will cause an additional (optional) AT entry action statement to be built.

There are many forced and recommended AT entries that require a CMD, REPLY, CODE, or USER action to be defined on the related message ID in the policy. Note

that no warning is issued if an action is defined for a message where a forced or recommended AT entry does not honor the action. See also "Other Forced AT Entries" on page 266.

AT Entries for SA z/OS Internal Messages

SA z/OS predefines automation for specific internal AOF, HSA, ING, EVE, EVI and EVJ messages and builds the corresponding AT entries. For non-predefined SA z/OS internal messages (for example, AOF*, HSA*, ING*, EVE*, EVI*, EVJ*), AT entries will be created that will not honor a CMD, REPLY, CODE, or USER action. These entries are created to avoid any interference with SA z/OS automation.

AT Entry Specialties

Defining message IEF403I, IEF404I, or IEF450I as a status message for a resource will generate an AT statement that contains a check for the job name in the AT statement condition.

Note: It is not recommended that you define the IEF403I message as a generic UP message under MVC, because this may cause the resource to be placed in an UP state at a time that is not accurate. Dependent resources may start too early and may fail.

AT Entries for Messages That Have Multiple Actions Defined

For certain messages there may be multiple actions defined for a single PDB entry or for several PDB entries. This influences the way ATs are built:

If an Override (OVR) action is defined for any policy entry, a corresponding AT entry is created at build time.

If there is an OVR action in conflict with a Type/Status Selection (AUTO action) for one message that is defined on several application instances, the AUTO action will cause a conflict warning to be issued at build time.

For messages *known* to SA z/OS, an AT entry is created as predefined by SA z/OS and not according to the AUTO, CMD, REPLY, CODE, or USER actions that may be defined.

For messages *unknown* to SA z/OS, then the behavior is as follows:

- If an AUTO action has been defined together with a CMD, REPLY, CODE, or USER action for the same message ID, then an AT entry is created honoring the AUTO action.

  Note: The generic routines (ACTIVMSG, HALTMSG, TERMMSG) check for optional commands or replies that are to be issued.
- If there is a CMD action defined together with a REPLY action for the same message ID, then an AT entry is created to issue a reply and then the command.
- If there is a CMD or REPLY action defined together with a CODE or USER action for the same message ID, then an AT entry is created to issue a command or reply.
- If there is a CODE action defined together with a USER action for the same message ID, or if there are only CODE or only USER actions defined for the same message ID, then no AT entry is built. If an AT entry is needed, then an override is required.

AT Entry Sequence

The sequence of AT entries for messages that are known to SA z/OS cannot be changed.

The location of wildcard niches (AT entries for unknown message IDs) cannot be changed.

AT entries in the same wildcard niche are sorted in a particular sequence, first by action and then by policy entry type. AT entries are created in order for the following actions:

1. OVR actions, then
2. AUTO actions, then
3. CMD or REPLY actions

Then the next level of sequencing within each of the above actions depends on the policy entry type:

1. APL instances, then
2. APL classes, then
3. MVS components (MVC)

Load

After the NetView automation tables have been generated using the customization dialog, they are ready to be loaded. INGAMS REFRESH can be used to refresh the complete SA z/OS configuration, that is, the Automation Manager Configuration (AMC), the agent's Automation Control Files (ACFs) and the related NetView Automation Tables (ATs) as they are defined in the SA z/OS Policy Database.

Alternatively, ATs can be loaded using ATLOAD.

Enabling Message Automation for the Automation Agent

READ authority *must* be given to AUTO1, AUTO2 and user tasks that will load the AT.

You can define those ATs in the PDB that are to be loaded by SA z/OS at initialization. Only those ATs defined in the PDB in entry type SYS, policy item SYSTEM INFO are refreshed.

Listing ATs

The DSILIST data set is used to store the AT listings, so if you want to view the listing of INGMSG01, issue the command:

```
br dsilist.ingmsg01
```

An AT listing is produced when SA z/OS loads an AT. You can use the advanced automation option (AAO) AOFMATLISTING to suppress listing by setting it to zero (see Appendix A, "Global Variables," on page 235).

The AT can be reloaded at configuration refresh (INGAMS, ACF ATLOAD)

Because of this you should:

- Use a separate DSILIST data set for each NetView
- Allocate the DSILIST data set as a PDSE in order to prevent Sx37 errors

A Guide to SA z/OS Automation Tables

NetView Automation Table Structure

SA z/OS provides a ready-to-use AT, INGMSG01. To activate the AT, perform the following steps:

1. Define the AT member INGMSG01 in the SYSTEM INFO policy of the system in the customization dialogs
2. Build the automation configuration files
3. Refresh the configuration using INGAMS REFRESH
4. Restart NetView with the new configuration

The SA z/OS AT contains:

- All entries for the SA z/OS basic automation
- Entries for subsystems and resources, such as MVS messages, JES2, JES3, OMVS, VTAM, TSO, NetView SSI, NetView Application, Automation Manager, SysOps, ProcOps, I/O Ops, SA z/OS Product Automation, OMEGAMON, RODM, GMFHS, TCP/IP, OMPROUTE, RESOLVER, ZFS, RMF, RMF Monitor III, VLF, DLF, LLA, APPC, ASCH, TWS, RACF, DFHSM, DFRMM, MQ, DB2, IMS, FDR, CICS, CMAS, IRLM, NFS Server, TPX (Terminal Productivity Executive), WebSphere, LDAP, etc.
- AT entries for messages that are defined in the PDB
- User include fragments

You do not have to customize this AT. All unused entries are disabled automatically according to the configuration that you use. If you want to have additional entries that are valid only for your environment, you can use either a separate AT (specified in the customization dialog) or use one of the user includes.

Figure 4 shows the structure of the AT:

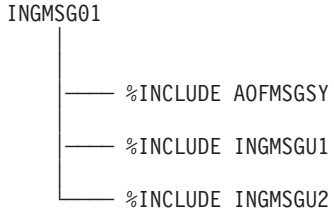


Figure 4. AT Structure

For information about how to use the INCLUDE fragments that SA z/OS provides, refer to "Using SA z/OS %INCLUDE Fragments" on page 31.

The following fragments are used by the AT:

Synonym Definitions
There is one fragment, AOFMSGSY, that is used to initialize the various synonyms used throughout the rest of the table. SA z/OS requires the synonyms to be suitably customized to reflect your environment.

SA z/OS Functional Definitions
These definitions (located in the fragment that is loaded as INGMSG02) contain automation table statements for specific functions of SA z/OS. You should not change these statements. Any modifications can be made in INGMSGU1.

Master Automation Tables

This section discusses the three master automation tables that SA z/OS provides.

INGMSG00: The automation table INGMSG00 is used for SA z/OS initialization. INGMSG00 should not have be modified by the user.

This table makes use of the synonyms defined in AOFMSGSY.

INGMSG01: INGMSG01 is suitable for use as a primary automation table.

INGMSG01 should not be included into any other table but should be activated as a separate table.

AOFMSGST: This is a table suitable for a NetView with a SA z/OS Satellite installed.

Integrating Automation Tables

If you have any user-written automation table statements that you still want to use, you must now combine your primary table with SA z/OS's. There are several approaches to achieve this.

Refer to the NetView documentation for more information on how to use NetView automation tables.

Multiple Master Automation Tables

Besides INGMSG01, you can specify multiple additional NetView automation tables for a system in the customization dialog. The tables are concatenated as entered in this panel and processed in this concatenation order.

You need not modify the INGMSG01 automation table or any of the fragments, except AOFMSGSY. It is easy to maintain SA z/OS automation table fragments. However, you have to watch for new messages. It is easy to maintain your entries, because they are independent from SA z/OS entries.

Using SA z/OS %INCLUDE Fragments

INGMSG01 is the master include member. It provides some message suppression that is necessary to prevent mismatches and duplicate automation before the first %INCLUDE.

The fragment INGMSGU1 can be used for user entries. These entries have precedence over the SA z/OS entries. The default INGMSGU1 is an empty member.

The fragment INGMSGU2 can be used for all entries that SA z/OS does not provide any entries for. The default INGMSGU2 is an empty member. During ACF COLD/WARM start the AT (or ATs) is (or are) loaded and write a listing to the DSILIST data set. This enables the use of the NetView AUTOMAN command to monitor and manage the AT (or ATs). Make sure that the size of your DSILIST data set is sufficient to store these listings. Without these listings you can just monitor/manage the ATs using AUTOTBL. It is recommended that you define your DSILIST data set as a PDSE so that regular data set compression is not required. Also you should make sure that the DSILIST DSN is unique to your NetView procedure.

An example output of AUTOTBL STATUS:

```
BNH361I THE AUTOMATION TABLE CONSISTS OF THE FOLLOWING LIST OF MEMBERS:
AUTO2    COMPLETED INSERT  FOR TABLE #1: INGMSG01 AT 04/16/02 19:34:59
AUTO2    COMPLETED INSERT  FOR TABLE #2: HAIMSG01 AT 04/16/02 19:35:00

IPSNO
BNH363I THE AUTOMATION TABLE CONTAINS THE FOLLOWING DISABLED STATEMENTS:
TABLE: INGMSG01 INCLUDE: __n/a___ GROUP   : INGCICS
TABLE: INGMSG01 INCLUDE: __n/a___ GROUP   : INGIMAGE
TABLE: INGMSG01 INCLUDE: __n/a___ GROUP   : INGIMS
TABLE: INGMSG01 INCLUDE: __n/a___ GROUP   : INGJES3
TABLE: INGMSG01 INCLUDE: __n/a___ GROUP   : INGOPC
```

An example of the AUTOMAN panel:

```
EZLKATGB              AUTOMATION TABLE MANAGEMENT

MEMBER    TYPE   LABEL/BLOCK/GROUP NAME(S)  STATUS    NUMBER OF STATEMENTS
--------  -----  -------------------------  --------  --------------------
INGMSG02  GROUP  INGCICS                    DISABLED           222
INGMSG02  GROUP  INGDB2                     ENABLED            120
INGMSG02  GROUP  INGIMAGE                   DISABLED           1
INGMSG02  GROUP  INGIMS                     DISABLED           107
INGMSG02  GROUP  INGJES2                    ENABLED            1
INGMSG02  GROUP  INGJES3                    DISABLED           1
INGMSG02  GROUP  INGOPC                     DISABLED           10
INGMSG02  GROUP  INGUSS                     ENABLED            1
```

In this example the configuration loaded does not use the IMS, CICS, OPC product automation and the IXC102A automation. It uses JES2, DB2 and USS automation.

Generic Synonyms: AOFMSGSY

This automation table fragment contains a number of synonyms that must be appropriately set. It is used in most master automation tables to set up the environmental parameters for the other fragments. The AOFMSGSY member is supplied by SA z/OS (in the SINGNPRM data set). You must customize it for each of your systems. The customized copy should be placed in the domain-specific data set for that system.

Note that many values in this table fragment are enclosed in triple single quotation marks. This means that the value of the synonym is the value entered surrounded by a single set of single quotation marks. This is necessary so that the value is treated as a literal and not an automation table variable.

Synonym	Usage and Default
%AOFALWAYSACTION%	This synonym contains the action statement used for all the messages within a Begin-End block that SA z/OS does not trigger any action for. **Default:** NULL The default is that *no* action will be taken and the message does not continue to search for further matches within the same AT.

Synonym	Usage and Default
%AOFDOM%	This synonym should contain the domain ID of the SA z/OS NetView on the system that it is automating. The synonym is used to screen messages to prevent the SA z/OS on this machine from reacting to a message that originated on another machine. If not set correctly, your automation will fail. **Default:** `&DOMAIN.` This is a default domain name used in a number of the samples.
%AOFSYS%	This synonym should contain the system name used in the last IPL of the system. It is used to screen messages to prevent the SA z/OS on this machine from reacting to events that have occurred on other machines. It is important if you are running on a JES3 global or in a sysplex with EMCS consoles. If not set correctly, your automation will fail. **Default:** `&SYSNAME.` This is a default system name used in a number of the samples.
%AOFSIRTASK%	NetView has a CNMCSSIR task that handles communications between the main NetView task and its SSI address space. This synonym should be set to the name of the task. If the synonym is not set properly, SA z/OS fails to initialize. **Default:** `&DOMAIN.SIR`
%AOFARMPPI%	This synonym should contain the name of the NetView autotask that is running the PPI interface from SA z/OS to z/OS. It is used to route commands from the NetView automation table to the autotask. **Default:** `AOFARCAT`
%AOFGMFHSWAIT%	The time interval SA z/OS waits after GMFHS initialization is complete before issuing the command to update the RODM with the current application automation states. Following the issuing of message DUI4003I GMFHS NETWORK CONFIGURATION INITIALIZED SUCCESSFULLY, GMFHS resets the color of all SA z/OS icons to grey (unknown). To set the SA z/OS icons' color to the current automation states after the initialization of GMFHS, SA z/OS must wait and issue the update command AFTER GMFHS has reset the colors to grey. **Default:** `00:02:00`

SA z/OS Message Presentation: AOFMSGSY

The presentation of SA z/OS messages (prefixed with AOF, ING, HSA, EVJ, EVE and EVI) under NetView® is controlled by the automation table. This uses a number of synonyms and task globals indicating your message display characteristics. The following synonyms determine the display characteristics for each type of message. There is one set for the normal presentation of the message (AOFNORM*x*) and a second set for the held presentation (AOFHOLD*x*).

Synonym	Usage and Default
%AOFHOLDI%	This synonym defines the actions taken for SA z/OS information (type I) messages that are held on your NCCF console. **Default:** `HOLD(Y) COLOR(GRE) XHILITE(REV)` This: • Ensures that the message is held • Causes the message to be displayed in reverse video green
%AOFHOLDA%	This synonym defines the actions taken for SA z/OS immediate action (type A) messages that are held on your NCCF console. As a rule, you should specify HOLD(Y) in the action. **Default:** `HOLD(Y) COLOR(RED) XHILITE(REV) BEEP(Y)` This: • Ensures that the message is held • Causes the message to be displayed in reverse video red • Sounds the terminal alarm when the message is displayed
%AOFHOLDD%	This synonym defines the actions taken for SA z/OS decision (type D) messages that are held on your NCCF console. As a rule, you should specify HOLD(Y) in the action. **Default:** `HOLD(Y) COLOR(WHI) XHILITE(REV) BEEP(Y)` This: • Ensures that the message is held • Causes the message to be displayed in reverse video white • Sounds the terminal alarm when the message is displayed
%AOFHOLDE%	This synonym defines the actions taken for SA z/OS eventual action (type E) messages that are held on your NCCF console. As a rule, you should specify HOLD(Y) in the action. **Default:** `HOLD(Y) COLOR(YEL) XHILITE(REV) BEEP(Y)` This: • Ensures that the message is held • Causes the message to be displayed in reverse video yellow • Sounds the terminal alarm when the message is displayed
%AOFHOLDW%	This synonym defines the actions taken for SA z/OS wait state (type W) messages that are held on your NCCF console. As a rule, you should specify HOLD(Y) in the action. **Default:** `HOLD(Y) COLOR(PIN) XHILITE(REV) BEEP(Y)` This: • Ensures that the message is held • Causes the message to be displayed in reverse video pink • Sounds the terminal alarm when the message is displayed

Synonym	Usage and Default
%AOFNORMI%	This synonym defines the actions taken for SA z/OS information (type I) messages that are not held on your NCCF console. As a rule, you should not specify HOLD(Y) in the action. **Default:** `COLOR(GRE)` This: • Ensures that the message is not held • Causes the message to be displayed in green
%AOFNORMA%	This synonym defines the actions taken for SA z/OS Immediate Action (type A) messages that are held on your NCCF console. As a rule, you should not specify HOLD(Y) in the action. **Default:** `COLOR(YEL) XHILITE(REV) BEEP(Y)` This: • Ensures that the message is held • Causes the message to be displayed in yellow • Sounds the terminal alarm when the message is displayed
%AOFNORMD%	This synonym defines the actions taken for SA z/OS Decision (type D) messages that are held on your NCCF console. You may find it beneficial to force these messages to be held. **Default:** `COLOR(WHI) XHILITE(BLI)` This: • Ensures that the message is held • Causes the message to be displayed in blinking white
%AOFNORME%	This synonym defines the actions taken for SA z/OS Eventual Action (type E) messages that are not held on your NCCF console. As a rule, you should not specify HOLD(Y) in the action. **Default:** `COLOR(YEL)` This: • Ensures that the message is not held • Causes the message to be displayed in yellow
%AOFNORMW%	This synonym defines the actions taken for SA z/OS Wait State (type W) messages that are held on your NCCF console. You may find it beneficial to force these messages to be held. **Default:** `HOLD(Y) COLOR(PIN) XHILITE(REV) BEEP(Y)` This: • Ensures that the message is held • Causes the message to be displayed in reverse video pink • Sounds the terminal alarm when the message is displayed

Operator Cascades: AOFMSGSY

The next set of synonyms defines a series of *operator cascades*. A cascade is basically a list of automation operators used in many of the fragments to route commands. If %CASCADE% is defined as a synonym for 'AUTMON AUTBASE AUTO1' and you route a command to it with ROUTE (ONE %CASCADE%) on an EXEC statement, the command is run on the first autotask in the cascade that is logged on. This provides you with a flexible, controllable means of providing backup processing tasks in case one of your normal tasks is unavailable.

Synonym	Usage and Default
%AOFLOPAUTOx%	This cascade defines the actions taken for SA z/OS information (type I) messages that are being held on your NCCF console. Given the number of informational messages that SA z/OS produces you may find it beneficial HOLD(N) to stop them from being held even if the user has asked for them to be held. **Default:** ''AUTOx''
%AOFOPAUTO1%	This cascade is used to route commands to AUTO1. If you have renamed AUTO1 you must change the synonym. **Default:** AUTO1 There is no backup for AUTO1. If it fails when it is needed, many other things will probably fail as well.
%AOFOPAUTO2	This cascade is used to route commands to AUTO2. If you have renamed AUTO2 you must change this synonym. **Default:** AUTO2 AUTO1 If AUTO2 is not active, AUTO1 does its work.
%AOFOPBASEOPER%	This cascade is used to send commands to BASEOPER. If you are not using the standard names for SA z/OS autotasks you must change this synonym. BASEOPER is mainly defined as a fallback operator and has very little work directly routed to it. **Default:** AUTBASE AUTO1 AUTBASE is the operator ID that SA z/OS uses for BASEOPER in its other samples. If AUTBASE is not active, AUTO1 is tried.
%AOFOPRPCOPER%	This cascade is used for XCF communication management. If you are not using the standard names for SA z/OS autotasks you must change this synonym. **Default:** AUTRPC AUTSYS AUTBASE AUTO1
%AOFOPSYSOPER%	This cascade is used to send commands to SYSOPER. If you are not using the standard names for SA z/OS autotasks you must change this synonym. SYSOPER is mainly defined as a fallback operator and has very little work directly routed to it. **Default:** AUTSYS AUTBASE AUTO1 AUTSYS is the operator ID that SA z/OS uses for SYSOPER in its other samples.

Synonym	Usage and Default
%AOFOPMSGOPER%	This cascade is used to send commands to MSGOPER. If you are not using the standard names for SA z/OS autotasks you must change this synonym. MSGOPER is mainly defined to respond to miscellaneous messages. **Default:** `AUTMSG AUTSYS AUTBASE AUTO1` AUTMSG is the operator ID that SA z/OS uses for MSGOPER in its other samples.
%AOFOPNETOPER%	This cascade is used to send commands to NETOPER. If you are not using the standard names for SA z/OS autotasks you must change this synonym. NETOPER is defined for VTAM automation. **Default:** `AUTNET1 AUTNET2 AUTSYS AUTBASE AUTO1` AUTNET1 and AUTNET2 are the operator IDs that SA z/OS uses for NETOPER in its other samples. NETOPER is the only sample automation function to have a backup defined in the samples.
%AOFOPJESOPER%	This cascade is used to send commands to JESOPER. If you are not using the standard names for SA z/OS autotasks you must change this synonym. JESOPER is mainly defined for JES automation. **Default:** `AUTJES AUTSYS AUTBASE AUTO1` AUTJES is the operator ID that SA z/OS uses for JESOPER in its other samples.
%AOFOPMONOPER%	This cascade is used to send commands to MONOPER. If you are not using the standard names for SA z/OS autotasks you must change this synonym. MONOPER is used for regular monitoring and subsystem startups. **Default:** `AUTMON AUTSYS AUTBASE AUTO1` AUTMON is the operator ID that SA z/OS uses for MONOPER in its other samples.
%AOFOPRECOPER%	This cascade is used to send commands to RECOPER. If you are not using the standard names for SA z/OS autotasks you must change this synonym. RECOPER is used for recovery processing. **Default:** `AUTREC AUTSYS AUTBASE AUTO1` AUTREC is the operator ID that SA z/OS uses for RECOPER in its other samples.
%AOFOPSHUTOPER%	This cascade is used to send commands to SHUTOPER. If you are not using the standard names for SA z/OS autotasks you must change this synonym. SHUTOPER coordinates automated shutdowns. **Default:** `AUTSHUT AUTSYS AUTBASE AUTO1` AUTSHUT is the operator ID that SA z/OS uses for SHUTOPER in its other samples.

Synonym	Usage and Default
%AOFOPGSSOPER%	This cascade is used to send commands to GSSOPER. If you are not using the standard names for SA z/OS autotasks you must change this synonym. GSSOPER is used for generic subsystem automation. **Default:** `* AUTGSS AUTSYS AUTBASE AUTO1` AUTGSS is the operator ID that SA z/OS uses for GSSOPER in its other samples. If you want to turn off the ″ASSIGN BY JOBNAME″ feature, that is, the advanced automation CGLOBAL variable *AOF_ASSIGN_JOBNAME* (see Appendix A, “Global Variables,” on page 235) has been set to 0, you must remove the asterisk (*), because this may cause serialization problems. **Note:** NetView’s ASSIGN-BY-JOBNAME command that occurs prior to the automation-table processing will only affect messages that are associated with an MVS job name.
%AOFOPWTORS%	This cascade is used to route commands concerning WTORS. If you are not using the standard names for SA z/OS autotasks you must change this synonym. Its use ensures that all WTOR processing is done on the same task and this is serialized. **Default:** `* AUTGSS AUTSYS AUTBASE AUTO1` This specifies that AUTSYS is to do all the WTOR processing.
%AOFOPGATOPER%	This cascade is used to route commands to this domain’s gateway autotask. Because the autotask name contains the domain ID you must modify this synonym. **Default:** `GATR`*`domain`*`.` AOF01 is the default domain used in the other samples. There is no backup as the gateway CLISTs expect to be running on GATOPER.

TEC Notification: AOFMSGSY

These synonyms are being used for notification of the Tivoli Enterprise Console (TEC).

Synonym	Usage and Default
%AOFTECTASKQ%	This is the name of the autotask for sending SA z/OS events to the Tivoli Enterprise Console (TEC) with quotes. **Default:** `''AUTOTEC''`
%AOFTECTASK%	This is the name of the autotask for sending SA z/OS events to the Tivoli Enterprise Console (TEC) without quotes. AOFTECTASK and AOFTECTASKQ must contain the same name (with and without quotes). **Default:** `AUTOTEC`

Synonym	Usage and Default
%AOFTECPPI%	This is the NetView PPI Receiver ID of the NetView message adapter (with quotes). **Default:** ''IHSATEC''
%AOFTECMODE%	Event generation mode (with quotes). Possible values are: **LOCAL** The NetView message adapter is running on this system. LOCAL is valid for the local configuration and for the focal point in the distributed configuration. **REMOTE** The NetView message adapter is running on a remote automation focal point. SA z/OS messages will be generated on this target system and forwarded to a remote automation focal point system. There is no local NetView message adapter that can process SA z/OS messages. REMOTE is valid for the target system in a distributed configuration. **Default:** ''LOCAL''

SA z/OS Topology Manager for NMC: AOFMSGST

These synonyms are used and defined in the AOFMSGST fragment.

Synonym	Usage and Default
%AOFOPTOPOMGR%	This is the name of the autotask that the SA z/OS topology manager runs on this system. **Default:** &DOMAIN.TPO
%AOFINITOPOCMD%	This is the command issued to initialize the SA z/OS topology manager. **Default:** INGTOPO INIT &DOMAIN.TPO
%AOFOPHB%	This is the name of the heart beat task needed on focal point. **Default:** AUTHB

Generic Automation Table Statements

The basic automation table contains a number of generic automation table entries that can reduce your automation table overhead considerably. These samples use some of the advanced features of SA z/OS to make automating your applications as simple and reliable as possible.

For some of these entries (IEF403I and IEF404I in particular) the message flow may be quite high. To handle this, you can insert additional entries in INGMSGU1 to suppress a block of messages. For example, if all your batch jobs started with the characters BAT or JCL, then the following entry would suppress them:

```
IF MSGID = 'IEF40'. & DOMAINID = %AOFDOM% THEN BEGIN;
*
  IF (TOKEN(2) = 'BAT'. | TOKEN(2) = 'JCL'.)
    THEN DISPLAY(N) NETLOG(N);
*
END;
```

Chapter 4. How to Monitor Applications

System Automation for z/OS provides different ways to monitor your applications:

- Using *observed status monitoring routines*, SA z/OS can determine whether your applications and several other automated resources are active, inactive, or in the process of being started. It is recommended to always enable observed status monitoring routines and to use the product-provided routines where possible. See "Observed Status Monitoring" for further details.
- With *monitor resources* you can optionally monitor the health of your applications and recover them on health status changes. SA z/OS distinguishes between active health monitoring and passive event-based health monitoring. See "Health Monitoring" on page 42 for further details.

Active and passive health monitoring is supported by SA z/OS in the following areas:

- Health monitoring of JES3, based on console messages
- Health monitoring of z/OS, DB2, CICS, IMS and other components, based on IBM Tivoli OMEGAMON II exceptions or IBM Tivoli OMEGAMON XE situations
- Health monitoring of CICS, based on CICSPlex® SM
- Health monitoring of IMS, based on console messages

Observed Status Monitoring

SA z/OS determines the observed status of an application by running a routine identified by the policy administrator in the customization dialog. The routine can be specified for an individual application (refer to *IBM Tivoli System Automation for z/OS Defining Automation Policy*), and a default monitor routine can be specified for all applications on an entire system (see the AUTOMATION INFO policy item in the customization dialog).

Table 2 lists the routines that can be specified as application monitors.

Table 2. Observed Status Monitor Routines

AOFADMON	This routine determines the status of an application by issuing the MVS D A, *jobname* command. The job name used is the job name defined in the customization dialog for the application. Possible values for the application monitor status as determined by this routine are Active, Starting, Inactive. IBM recommends to use INGPJMON instead of AOFADMON.
AOFATMON	This routine is used to determine the status of a task operating within the NetView environment.
AOFAPMON	This routine determines the status of a program-to-program interface (PPI) receiver.
AOFCPSM	This routine is a dedicated routine used to monitor the status of the SA z/OS processor operations applications.
AOFNCMON	This routine is used to determine the status of the NETCONV connection running between the NMC server and NetViewfor z/OS.

Table 2. Observed Status Monitor Routines (continued)

AOFUXMON	This routine determines the status of a resource with application type USS. This resource can either be a z/OS UNIX process, a file system in the UNIX file system (HFS), or a TCP port. Depending on the nature of the resource (process, file, or port) AOFUXMON decides which internal monitoring method to use.
INGPJMON	This routine determines the status of an application by searching z/OS for address spaces with a particular job name. The job name used is the job name defined in the customization dialog for the application.
INGMTSYS	With this routine, IMAGE applications for BCPII usage can be monitored.
INGPSMON	This routine monitors the subsystem's registration to the subsystem interface.
ISQMTSYS	With this routine, a processor operations target system resource represented by its proxy can be monitored. See "Automating Processor Operations Resources of z/OS Target Systems Using Proxy Definitions" on page 83 for examples of how to use a proxy definition. Active operator console connections are mandatory and will be used for sending a z/OS command (for example, d t) and receiving the related response.

SA z/OS expects certain return codes from all monitor routines, either from SA z/OS provided ones or from your own routines. These can be one of the following:

RC	Meaning
0	Active
4	Starting
8	Inactive
12	Error

Health Monitoring

Overview

Health monitoring is accomplished using special resources called *monitor resources*. Monitor resources, which have a resource type MTR, are policy objects that are used to obtain the health status of other resources, typically applications or application groups, or more generally, any object that can be monitored. The health status is useful when you need to know how well a resource is performing and not simply that it is active.

The health status can be used to provide application-specific performance and health monitoring information, for example, an application may be active but it is failing to meet performance objectives defined by the system administrator. The health status can be used either for information only, or by the automation manager to make decisions and, if necessary, trigger automation for the application.

Monitor resources are defined in the customization dialog with entry type MTR. They are resources with similar characteristics as all other SA z/OS resources. Monitor resources are started and stopped using the INGREQ command and can have a service period defined for them.

Monitor resources are connected to application resources (APLs) or application group resources (APGs). The health status of the monitored object is propagated to the APLs and APGs and results in a combined health status there. You can define and connect MTRs in the customization dialog (see *IBM Tivoli System Automation for z/OS Defining Automation Policy*).

Monitor resources obtain the health status of an object in two different ways:

- Actively, by polling—that is executing a monitoring command periodically
- Passively, by processing events

Active monitors are scheduled periodically based on the interval defined in the MTR policy.

Passive monitors do not have a monitor interval but can have a monitor command defined for them for initial health status determination. They rely on other events to set the health status using the INGMON command.

Monitor resources can be explicitly bound to the object that they are monitoring and optionally to a job. This allows SA z/OS to handle a variety of monitoring events in a generic way. A monitored object can be, for example, an OMEGAMON XE situation, or an event posted by CICSPlex System Manager (CICSPlex SM). See "Passive, Event-Based Health Monitoring" on page 46. Note that the monitored object is derived from the monitor resource name, if none was specified.

There can be one or more recovery commands associated with each health status (NORMAL, WARNING, MINOR, CRITICAL and FATAL). These commands are invoked by SA z/OS when the monitor resource switches to the corresponding health status.

You can display and control monitor resources with the DISPMTR command. Monitor resources are also displayed on the Tivoli Enterprise Portal (TEP) as well as SDF and NMC, provided that the appropriate inform list specifications have been made (see "Step 5: Decide Where to Forward Subsystem Information To" on page 2)

Monitor Resource Commands

When defining a monitor resource you can specify activate, deactivate and monitor commands. Any command is suitable that can be executed in the NetView environment. These commands are divided into two groups:

- NetView activate and deactivate commands that expect a return code of zero
- Monitor commands that return a health status

The main difference between these two groups is that the activate and deactivate commands are executed only once, and SA z/OS expects a return code of zero.

If the activate command ended with a non-zero return code, the monitor resource remains in an INACTIVE status. The monitor resource ends in a BROKEN status if the deactivate command ended with a non-zero return code.

- The activate command is optional and can be used to establish the environment the monitoring routine can run in. The command is executed every time the monitor is started. The command must exit with return code 0.
- The deactivate command is optional and can be used to cleanup the environment. The command is executed every time the monitor is stopped. The command must exit with return code 0.

- The monitor command is executed after the activate command and then periodically if a monitoring interval is given. SA z/OS expects the monitor command to return a valid health status code. Additionally the monitor command can issue a message that is then attached to the health status. The absence of a monitoring interval indicates that the given monitor resource is a passive or event-based health monitor. In this case, the monitor command is optional and, if specified, it is invoked for initial health monitoring only. Otherwise, if a monitoring interval is provided, the given monitor resource is an active health monitor. In this case, a monitor command must be provided to return a health status.

The activate, deactivate and monitor command can be a command procedure written in any language that is supported by NetView: REXX, Assembler, PL/I, C, or the NetView Command List Language (NCLL). Writing a monitor routine can be simple or it can be complex. The complexity depends upon the application that you are attempting to monitor.

Writing a Recovery Routine

The recovery routine is invoked every time the monitor resource switches to the health status that the recovery routine is defined for. The goal of the recovery routine is to bring the monitor resource, and thus the monitored object, back to a health status of NORMAL.

Recovery Techniques

User data in the MESSAGES/USER DATA policy item can be used to disable additional recovery processing while other recovery is already in progress. In combination with the predefined keyword DISABLETIME, the recovery disable time can be specified in the formats hh:mm:ss, mm:ss, :ss, or mm. While recovery is disabled, no commands are processed on behalf of this monitor resource for messages and exceptions that are specified in the MESSAGES/USER DATA policy item.

Recovery is automatically enabled after the recovery disable time has expired. Recovery can also be enabled prematurely by calling the generic routine INGMON with the option CLEARING=YES, for example:

```
INGMON CI2XREP MSGTYPE=XREP CLEARING=YES
```

In some cases, it is necessary to force increasingly strong recovery actions over a period of time. This can be accomplished using a PASS count that starts at 1 and runs to 99. SA z/OS maintains the PASS count individually per message or exception, and increments the PASS count each time that message or exception is processed. Upon successful recovery, it is the installation's responsibility to reset the PASS count. When specified with option CLEARING=YES, INGMON enables command processing for messages and exceptions, and resets the PASS count.

Task Global Variables for Recovery Routines

The following task global variables can be accessed by the recovery routine:

Task Global Variable	Value
&EHKVAR1	Contains the monitor name
&EHKVAR2	Contains the current health status
&EHKVAR3	Contains the old health status
&EHKVAR4	Contains the message that is associated with the health status

Active Health Monitoring

In general, the monitor command will need to issue one or more commands to generate data, process the data, and set a return code. The return code is then used by SA z/OS to determine the health status for the resource. The possible return codes and the corresponding health status are given in Table 3.

Table 3. Health Status Return Codes

Return Code	Health Status	Description
1	BROKEN	The monitor detected an unrecoverable error. SA z/OS will stop monitoring.
2	FAILED	The monitor is currently unable to obtain a health status. SA z/OS will keep the monitor active because the problem might disappear.
3	NORMAL	The monitor detected normal operation of the monitored object.
4	WARNING	The monitor detected a certain degree of degradation in the operation of the monitored object.
5	MINOR	The same as WARNING, but more severe.
6	CRITICAL	The same as MINOR, but more severe.
7	FATAL	The same as CRITICAL, but more severe.
8	DEFER	Used internally.

The health status values (with the exception of UNKNOWN) will affect the compound status in the automation manager.

Most monitor commands will use UNKNOWN, NORMAL, and WARNING statuses. The MINOR, CRITICAL, and FATAL statuses can be used as gradients to indicate that a problem is getting worse. BROKEN and FAILED are statuses that describe the status of the monitor itself and may be seen if an error is encountered with the monitor command.

Optionally, the monitor routine can issue a message describing the condition that will be trapped by the SA z/OS process that invoked the monitor. The message can be viewed on the DISPMTR panel.

Every monitor command will need several basic steps:

1. Issue one or more commands to collect data and interrogate the results.
2. Based on the results from the command or commands, set the return code to a value from 1 through 8 and, optionally, perform processing based on that value.
3. Optionally, supply more descriptive information about the health status in a message that can be viewed with the DISPMTR command.
4. Exit with the return code so SA z/OS can set the health status appropriately.

Figure 5 on page 46 is an example using the NetView PING command within a PIPE to query the status of a TCP/IP stack on a remote system. The IP address is passed on input. The routine uses the average round trip time (RTT) for the request provided in message BNH770I to determine the health.

```
/*REXX MYMON */
Arg parm
monrcs='BROKEN FAILED NORMAL WARNING MINOR CRITICAL FATAL DEFER'
'PIPE (STAGESEP | NAME PING)',
'| NETV PING' parm,
'| LOCATE 1.8 /BNH770I /',
'| STEM out.'
if out.0 = 0 then
  lrc = wordpos('FATAL',monrcs)
else
  do
    parse var out.1 . 'averaging' ms 'ms' .
    say 'PING lasted' ms 'ms'
    select
      when ms < 10 then lrc = wordpos('NORMAL',monrcs)
      when ms < 20 then lrc = wordpos('WARNING',monrcs)
      when ms < 30 then lrc = wordpos('MINOR',monrcs)
      when ms < 40 then lrc = wordpos('CRITICAL',monrcs)
      otherwise lrc = wordpos('FATAL',monrcs)
    end
  end
Return lrc
```

Figure 5. Sample Monitor Command

Passive, Event-Based Health Monitoring

Overview

Passive, event-based monitoring allows you to react to events, for example a message, an OMEGAMON XE situation, or a CICSPlex SM event, directly. In contrast to active health monitoring, SA z/OS does not have to query the monitored object status periodically but is informed only when such an event has occurred.

The definitions in the MONITOR INFO policy item for a monitor resource allow you to define an object that the monitor resource is bound to and optionally a job that the monitor resource accepts events from.

The **Monitored Object** specification for the monitor resource can follow any naming convention that might be required for the monitoring process. For example, for CICS monitoring it has the prefix CPSM, followed by the CICS name, the type (such as a connection), and the name. For a link called CT12, the monitored object is called as follows, for example:

`CPSM.CICSTOR1.CONNECT.CT12.`

Whereas for monitoring OMEGAMON XE situations, it has the prefix ITM, followed by the situation name, for example: ITM.MYAUXSHORTAGE_WARN.

There can be only one monitored object per monitor resource but more than one monitor resource can be bound to a monitored object, for example, several IMS monitors might specify OLDS as an object.

You can also optionally specify the **Monitored Jobname** that a monitor resource accepts events from. Thus, for example in the case of IMS monitor resources, you might specify a job name of IMS1 for monitor resource MTR1 and IMS2 for MTR2. If an event arrives for OLDS and the issuer is IMS1 only MTR1 is affected.

Event Types

In the simplest case, an event is represented by a plain message issued by a job. All monitor resources that register for a particular message will accept this message unless you also specified the monitored jobname.

In other cases, for example for OMEGAMON XE situations or events reported by CICSPlex SM, the event is represented by a triggering message provided by SA z/OS for the purpose of health monitoring only. This message, ING150I, that contains the monitored object name or the job can then be used by SA z/OS to locate the monitor resource and to set the health status or issue commands. This allows SA z/OS to handle a variety of monitoring events.

INGMON, the generic routine that is responsible for health monitoring, is invoked from the NetView automation table whenever ING150I or any other message a monitor resource has registered for is issued. It locates the monitor resource for a given monitored object or job and then looks up the code match table for the health status or commands, or both, that should be issued whenever the triggering event occurs.

Code Matching for Event-Triggering Messages

INGMON allows you to pass up to three codes that, when specified, are used to determine a specific set of commands to be issued in case of an event-triggering message. For message ING150I, SA z/OS creates an automation table entry where **Code 1** is used to select commands by event severity. For other messages, you can override the default automation table entry and pass the appropriate tokens in **Code 1**, **Code 2**, and **Code 3**, as you require.

In any case, the **Value Returned** field contains one or two tokens separated by a blank. The first token is a required command selector that can be one of the following:

selection
: Execute commands with the given selection or commands for which no selection is specified.

#
: Perform pass processing and execute all commands that match the current pass.

#*selection*
: Interpret *selection* as another pseudo message ID. Perform pass processing for this message and execute all commands that match the current pass.

  This is useful for pass processing on behalf of the event triggering message, for example, ING150I. Suppose you have one entry for WARNING and one for CRITICAL. When you do pass processing for ING150I your pass counter may be on 5, for example, when the first CRITICAL event comes in (because you already had 4 WARNING events).

  However, with #*selection* you can specify, for example, a value returned of #MYWARN WARNING and #MYCRIT CRITICAL for the corresponding levels. INGMON will do pass processing for the pseudo-message MYWARN and set the health status WARNING for a WARNING event. For a CRITICAL event it will do independent pass processing for the pseudo-message MYCRIT and finally set a health status of CRITICAL.

  Remember to set the IGNORE action for the pseudo-messages to avoid AT entries being built.

The second token in the **Value Returned** column of the Code Processing panel indicates the optional health status to be set. If specified, it must be separated by a blank from the selection criterion.

Programming Techniques

Commands that are called by INGMON have access to the message that triggered the invocation using the NetView SAFE, AOFMSAFE, for example:

```
/* MYCLIST, called by INGMON */
'PIPE SAFE AOFMSAFE | STEM MSG.'
If msg.0 > 0 Then
  msgtext = msg.1        /* first message line */
```

In addition, INGMON fills the task global variables &EHKVAR0, &EHKVAR1-9, and &EHKVART with tokens that are derived from the message or exception that INGMON was invoked by. For messages, the assignment starts with the message ID, and for exceptions, it starts with the exception ID. The following two examples illustrate how message and exception tokens are assigned to these task global variables.

Example 1:

```
$HASP9211 JES MAIN TASK NOT RUNNING. DURATION- hh:mm:ss.xx
```

Task Global Variable	Value
&EHKVAR0	$HASP9211
&EHKVAR1	JES
&EHKVAR2	MAIN
&EHKVAR3	TASK
&EHKVAR4	NOT
&EHKVAR5	RUNNING.
&EHKVAR6	DURATION-
&EHKVAR7	hh:mm:ss.xx
&EHKVAR8 &EHKVAR9 &EHKVART	*NULL*

Example 2:

```
ING080I CI2XREP/MTR/KEYA OMSY4MVS OMIIMVS XREP Number of Outstanding Replies = 4
```

Task Global Variable	Value
&EHKVAR0	XREP
&EHKVAR1	Number
&EHKVAR2	of
&EHKVAR3	Outstanding
&EHKVAR4	Replies
&EHKVAR5	=
&EHKVAR6	4

Task Global Variable	Value
&EHKVAR7 &EHKVAR8 &EHKVAR9 &EHKVART	*NULL*

When defining commands to be issued by the INGMON generic routine, the &EHKVAR*x* variables can be used to be replaced by the corresponding tokens of the message or exception.

When INGMON looks up the monitor resource for a given monitored object or job name, or both, it is possible to skip monitor resource processing dynamically through a user-specified REXX expression. In the absence of such a REXX expression, INGMON locates the monitor resource with the given monitored object name for the job that issued the message and proceeds with health status setting and commands as defined in the automation policy. By adding a REXX expression to the User Defined Data panel within the MESSAGES/USER DATA policy item for the automated message, further processing can be disabled depending on the result of this REXX expression.

To do this, the predefined keyword INGMON_FUNCTION is specified as a keyword and an arbitrary REXX expression is defined as the value in the User Defined Data panel. If the result of the REXX expression is false (that is, 0), processing is stopped, otherwise INGMON processing continues. The following example controls monitor resource processing, based on the day of week that happens to be defined in the common global variable DAY_OF_WEEK. Processing continues only if the current day is not a Sunday:

```
Entry Name : MYMTR                Message ID : ING150I

To change keyword-data pair, specify the following:

Keyword
Data
INGMON_FUNCTION
cglobal('DAY_OF_WEEK') \= 'SUN'
```

When a monitor resource is defined with a monitor command but without an interval, the initial health status of such a passive monitor resource is obtained at monitor resource start time only. Any other health status update must be derived from events that the monitor resource has registered for.

It is however possible to issue the monitor command at any point in time by executing the command AOFRCMTR. This command expects the monitored object name and optionally a job name as parameters. It locates the corresponding monitor resource and, if specified, issues the monitor command.

See *IBM Tivoli System Automation for z/OS Programmer's Reference* for the syntax of AOFRCMTR.

Health Monitoring using OMEGAMON

SA z/OS allows you to interact with IBM Tivoli OMEGAMON II and IBM Tivoli OMEGAMON XE products to collect key performance indicators that represent the health status of address spaces, middleware, or even the system. The following sections show you how to interact with these products using monitor resources.

Overview

The SA z/OS OMEGAMON interface lets you gather a wide range of performance data on a system. You can gather data from the following performance monitoring products:

- IBM Tivoli OMEGAMON II for MVS
- IBM Tivoli OMEGAMON II for CICS
- IBM Tivoli OMEGAMON II for IMS
- IBM Tivoli OMEGAMON II for DB2
- IBM Tivoli OMEGAMON XE products
- Other IBM Tivoli Monitoring products running on z/OS

Exception analysis is an OMEGAMON feature that monitors predefined *thresholds* in a system. Each time exception analysis is invoked, an exception is displayed on the OMEGAMON console if a threshold is exceeded. Using SA z/OS, you can then act on these exception alerts by running execs or issuing commands, including issuing commands back to the host OMEGAMON.

Situations are much like exceptions but they are based on a combination of logical expressions and even on the status of other embedded situations. Each product based on the IBM Tivoli Monitoring infrastructure, such as IBM Tivoli OMEGAMON XE, provides a set of predefined situations that you can use as is, or modify as you wish. You can also create your own situations to tailor the monitoring to your specific needs. Situations are edited and displayed on the Tivoli Enterprise Portal (TEP). Using a TEP function called Reflex Automation, you can inform SA z/OS about a particular situation and then act upon it.

IBM Tivoli Monitoring services also allow you to interact with each and every product based on this infrastructure through a standardized SOAP services interface on the Tivoli Enterprise Monitoring Server (TEMS). SOAP services exist, for example, to obtain data from a particular object collected by Tivoli OMEGAMON XE for z/OS. Other services allow you to automatically manage situations and TEP workflow policies, or to send universal messages to the universal message console.

You can set up monitor resources to:

- Monitor sets of exceptions that may be of interest using an active monitor resource and set an application's health status based on the existence of such exceptions
- React to and resolve conditions that cause those exceptions
- Monitor sets of situations that may be of interest using a passive monitor resource, set an application's health status and react to and resolve conditions that cause those situations

Assumptions

Various topologies are possible for SA z/OS with IBM Tivoli OMEGAMON II monitors and IBM Tivoli Monitoring products such as OMEGAMON XE:

- There can be one or more monitoring product per system
- Connectivity is through VTAM and the NetView Terminal Access Facility (TAF) for OMEGAMON II and through TCP/IP for OMEGAMON XE
- A TEMS SOAP Server is running locally, on a remote system or on a distributed system
- SA z/OS can act as a focal point either:
  - Globally, monitoring data from monitoring products running on different systems
  - Locally, monitoring data from monitoring products running on the local system

The following assumptions are made about the topologies that can be adopted for interaction with OMEGAMON II:

1. The OMEGAMON product is installed on each system where MVS and CICS, DB2, or IMS is installed.
2. OMEGAMON monitors are installed and configured already to support multiple VTAM-based connections to it. For interoperability with SA z/OS, logical units of type 3270 model 2 (24x80) are required.
3. OMEGAMON monitors are setup to interact with an external security product such as IBM SecureWay® Security Server for z/OS (formerly RACF).
4. OMEGAMON exceptions are reported when the threshold that is defined in OMEGAMON is exceeded. That threshold must be agreed within an installation because it must cater for the least severe condition that there might be an alert for.

The following assumption is made regarding the interaction with OMEGAMON XE:

1. Reflex automation is executed on the OMEGAMON XE agent that created the corresponding situation event

OMEGAMON Interaction

The following subsections assume that, for OMEGAMON II interaction, you have defined one or more OMEGAMON sessions and automated functions that are designated to handle network communication using the SA z/OS customization dialog. For details on defining OMEGAMON sessions, refer to the OMEGAMON SESSIONS and AUTHENTICATION policy items in the Network (NTW) entry type and to the OPERATORS policy in the Auto Operators (AOP) entry type described in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

For OMEGAMON XE interaction using SOAP services you have to specify each SOAP server in the automation policy that you want to connect to. For details on defining SOAP servers, refer to the SOAP SERVER policy item in the Network (NTW) entry type described in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Using the INGOMX Programming Interface

INGOMX acts as the interface between operators (or auto-operators) and OMEGAMON. This includes not only any of the classic OMEGAMON monitors for CICS, DB2, IMS, and MVS, but also OMEGAMON XE monitors and other IBM Tivoli Monitoring products running on z/OS.

For the classic OMEGAMON monitors, INGOMX can be used to issue OMEGAMON major, minor, and immediate commands, and to filter one or more

exceptions of interest from the list of exceptions reported by OMEGAMON exception analysis. Each request is written to the console (but not exposed to NetView) in the format as produced by the OMEGAMON monitor. When exception filtering is requested, multiple exception lines for one exception are combined into a single line and written to the console as a single message if the filter criterion (XTYPE) matches. INGOMX is best used within a NetView PIPE.

The INGOMX SOAP interface allows you to issue any of the SOAP services supported by the TEMS SOAP server, for example to

- Obtain attributes of interest from a particular OMEGAMON XE object, for example, Job_name and CPU_percent from the OMEGAMON XE for z/OS object Address_Space_CPU_Utilization
- Start and stop situations as well as TEP workflow policies
- Issue a universal message
- Send an event into the IBM Tivoli Monitoring platform

The full set of SOAP services and a description of the XML-syntax is described in *IBM Tivoli Monitoring Administrator's Guide*.

The following examples illustrate the use of INGOMX. They are based on an OMEGAMON for MVS session with the name OMSY4MVS. The same techniques also apply to other OMEGAMON monitors. For more details, refer to *IBM Tivoli System Automation for z/OS Programmer's Reference*.

Example 1. Returning Information on Common Storage Utilization Using the CSAA Command:

```
INGOMX EXECUTE,NAME=OMSY4MVS,CMD=CSAA
| IPXNG     CSAA   SUMMARY
| IPXNG    +
| IPXNG    +           System
| IPXNG    +          Maximum  Pre-CSAA   Orphan       Usage
| IPXNG    +          -------  --------  -------  ---------------0___2___4___6___8___100
| IPXNG    +    CSA     3312K    1247K         0    1247K   37.6%|------>              |
| IPXNG    +   ECSA   307740K   78797K         0   78797K   25.6%|---->                |
| IPXNG    +    SQA     1620K     660K         0     660K   40.8%|------->             |
| IPXNG    +   ESQA   145696K   23930K         0   23930K   16.4%|-->                  |
```

Example 2. Using OMEGAMON Command Modifiers:

```
INGOMX EXECUTE,NAME=OMSY4MVS,CMD=ALLJ,MOD=#
| IPXNG    #ALLJ       166
INGOMX EXECUTE,NAME=OMSY4MVS,CMD=ALLJ,MOD=<
| IPXNG    <ALLJ  *MASTER*   PCAUTH     RASP    TRACE  DUMPSRV    XCFAS      GRS  SMSPDSE+
| IPXNG    +       CONSOLE      WLM  ANTMAIN ANTAS000     OMVS IEFSCHAS   JESXCF  ALLOCAS+
| IPXNG    ...
```

Example 3. Trapping Outstanding Operator Replies:

```
INGOMX TRAP,NAME=OMSY4MVS,XTYPE=(XREP)
| IPXNG    + XREP Number of Outstanding Replies = 5
```

Example 4. Issuing OMEGAMON Minor Commands:

```
/* REXX-Routine EXMINOR */
cmd.1 = "CMD=SYS"   /* Major command, issued ahead of its minors   */
cmd.2 = "CMD=FCSA"  /* Minor: CSA frames below 16M                 */
cmd.3 = "CMD=FCOM"  /* Minor: CSA, LPA, SQA, and nucleus below 16M */
cmd.0 = 3
'PIPE STEM cmd. COLLECT',
'| NETV INGOMX EXECUTE,NAME=OMSY4MVS,CMD=*',
'| CONSOLE ONLY'
* IPXNG    EXMINOR
| IPXNG     SYS    >> WLM Goal mode OPT=00 SYSRES=(150526,8812) <<
| IPXNG     fcsa       328     1312 K
| IPXNG     fcom       849     3396 K
```

There is no need to explicitly establish a session between an operator and a particular OMEGAMON monitor before using INGOMX; such sessions are established automatically on their first use.

Selective protection of individual OMEGAMON sessions and commands, or both, is possible based on the NetView Command Authorization Table. Details can be found in the appendix, “Security and Authorization”, in *IBM Tivoli System Automation for z/OS Planning and Installation*.

To use a SOAP service, for example to obtain certain attributes from an OMEGAMON XE object, you first have to describe the request’s parameters in the form of an XML document. The XML document is validated and rejected by the SOAP server if it is found to be incorrect or incomplete. The spelling of the names enclosed in ’<’ and ’>’ is significant because XML is a case-sensitive document description language. Also, because the structure of every XML document is hierarchical, each element must be enclosed by an opening name (for example, ’<CT_Get>’) and a corresponding closing name denoted by a forward slash preceding the name (for example, ’</CT_Get>’).

The following is an example that describes the request parameters to retrieve the Job_Name, the address space ID (ASID), and the CPU_Percent attributes from the OMEGAMON XE for z/OS object, Address_Space_CPU_Utilization, for all jobs with a CPU percentage greater than 1.0. In this example, the object that has been queried is collected on the TEMS called KEYAS:CMS.

```
<CT_Get>
  <target>KEYAS:CMS</target>
  <object>Address_Space_CPU_Utilization</object>
  <attribute>Job_Name</attribute>
  <attribute>ASID</attribute>
  <attribute>CPU_Percent</attribute>
  <afilter>CPU_Percent;GT;10</afilter>
</CT_Get>
```

You can pass this XML document either by pointing INGOMX to a sequential or partitioned data set, or in the default SAFE, assuming INGOMX is invoked in a NetView PIPE.

When INGOMX is invoked, the SOAP server that is connected to must be specified. In the following example, it is assumed that you have defined a SOAP server called KEYAYA in the SOAP SERVER policy item of the Network (NTW) entry type using the SA z/OS customization dialog. This definition includes the host name or IP address, the SOAP server's port and the path name of the SOAP service. The request parameters as shown above are located in the member GETCPU in the partitioned data set SYS1.SOAP.DATA:

```
soapds = 'SYS1.SOAP.DATA(GETCPU)'
soapsrv = 'KEYAYA'
Address NETVASIS 'PIPE (END % NAME GETCPU)',
'| NETV (MOE) INGOMX SOAPREQ SERVER='soapsrv' DATA='soapds,
'| L: LOC 1.8 'd||'DWO369I '||d,
'| EDIT SKIPTO 'd||'RETURN CODE'||d,
'       UPTO 'd||'.'||d,
'       WORD 3 1',
'| VAR omx_rc',
'%L:',
'| CON ONLY'
```

On the successful return of INGOMX, the output of the SOAP server is returned in the multiline ING160I message:

```
ING160I RESPONSE FROM SOAP SERVER: 9.xxx.xxx.xxx:1920///cms/soap
Job_Name:ASID:CPU_Percent
IXGLOGR:20:2.1
NET:59:2.1
RMFGAT:89:6.9
SDM1IRLM:108:1.7
BBOS001S:113:22.1
YANAMSJH:117:3.9
```

The first row of this message documents the IP address of the SOAP server that responded, that is, KEYAYA in the example (IP address anonymized).

The second row describes the names of the attributes returned by the SOAP server. The attribute names are separated from each other by the non-printable character X'FF' (represented by a `:`).

The third and all following rows contain the actual data that has been requested. The attribute values are presented in the same sequence as the corresponding attribute names in the second row. Also, like the attribute names, the attribute values are separated from each other by the non-printable character X'FF' (represented by a `:`).

The tabular structure of this message allows you to easily process it in a NetView PIPE.

Using the INGMTRAP Monitor Command

INGMTRAP is a customized interface to INGOMX that provides filtering capabilities for exceptions of interest as reported by OMEGAMON exception analysis and triggering of automation on behalf of such exceptions. For each exception that matches the XTYPE filter that is provided by the caller, INGMTRAP issues message ING080I, which is exposed to NetView. For example:

```
ING080I CI2XREP/MTR/KEYA OMSY4MVS OMIIMVS XREP Number of Outstanding Replies = 4
```

If no exception matches the XTYPE filter that is provided by the caller, INGMTRAP creates a ING081I message that is not exposed to NetView but written to the monitor resource's log to document that no exception has been found. For example:

```
ING081I CI2XREP/MTR/KEYA OMSY4MVS OMIIMVS NO EXCEPTION FOUND
```

INGMTRAP can only be used as a monitor command. This means that it has to be specified directly as a monitor command in the definition of a monitor resource, or it has to be called on behalf of such a monitor command. The following example illustrates what you need to specify on the MONITOR INFO policy in entry type monitor resource (MTR) in order to trap outstanding operator replies that are reported by OMEGAMON for MVS session OMSY4MVS:

```
INGMTRAP NAME=OMSY4MVS,XTYPE=(XREP)
```

Be careful when specifying a list of exceptions: each exception may cause an ING080I message to be issued. Because each occurrence of an ING080I message will trigger health status processing of the monitor resource, make sure you understand the impact that this may have on the monitor resource's final health status.

For more details about INGMTRAP refer to *IBM Tivoli System Automation for z/OS Programmer's Reference*. For more details about defining monitor resources, refer to *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Health Monitoring Based on OMEGAMON Exceptions

This section describes how to set up the monitor resources for health-based monitoring based on OMEGAMON exceptions using the customization dialogs, provides a sample scenario, and gives recommendations when using OMEGAMON in combination with monitor resources.

Defining the Monitor Resources

By combining monitor resources and the OMEGAMON interaction methods described in "OMEGAMON Interaction" on page 51, automation can be triggered as a result of analyzing the output reported by OMEGAMON and by the setting of an appropriate health status.

OMEGAMON exceptions can be periodically monitored using a monitor resource and the monitor command INGMTRAP. There are a variety of ways to handle such exceptions:

1. In the customization dialog, the MESSAGES/USER DATA policy of a given monitor resource needs to state the health status of each exception that INGMTRAP has been set up to monitor. Unlike messages, OMEGAMON exceptions are denoted by a '+' sign, followed by a blank and then a 4-character OMEGAMON exception ID.
2. In addition to the health status, a series of one or more commands can be specified to handle that particular exception. Commands are processed in the same way as for any other resources that a MESSAGES/USER DATA policy is provided for, such as applications (APL). This includes escalation processing based on a PASS count, or processing based on a selection value that can be defined using CODEs that are derived from a message.
3. The HEALTHSTATE policy can be used to issue recovery commands on behalf of an OMEGAMON exception each time the health status changes.

No matter which method or combination of method are chosen, the process of handling an exception is triggered by the occurrence of an ING080I message for a particular monitor resource and exception. The automation table that is built from the definitions in the MESSAGES/USER DATA policy contains statements that invoke the generic routine INGMON to set the monitor resource's health status and to issue commands in response to exceptions. In most cases, the necessary

entries in the NetView Automation Table are created automatically by SA z/OS. In some rare cases when, for example, command selection should be based on CODEs, it is necessary to override the automation table definition of the exception, and to specify up to 3 codes (CODE1, CODE2, and CODE3) on the invocation of INGMON.

Alternatively, an installation-written monitor command can be used to issue INGOMX for a series of exceptions to one or more OMEGAMON monitor. Such a monitor command then returns with an appropriate health status that is based on the analysis of the output produced by INGOMX. The recovery commands that are issued when the health status changes are specified in the HEALTHSTATE policy of that monitor resource.

Example Scenario

To illustrate how SA z/OS and OMEGAMON operate together, consider the following scenario.

> Suppose there is a DB2 application that should be continuously monitored. Of particular interest is the availability of primary active logs. The LOGN exception indicates that fewer primary active logs exist than specified by the respective threshold value. This is considered a critical health indicator because it can cause a DB2 hang situation if the last primary active log becomes 100% full. Such a situation can only be resolved by making one or more additional primary active logs available again.

In order to monitor this situation and react accordingly, the automation policy has to be changed. First, define the session attributes for the OMEGAMON for DB2 monitor, if they do not yet exist, to be able to establish a VTAM connection. The OMEGAMON session is referred to by its *session name*. Then review the number of session operators (automation operators) that will be started to handle the VTAM session traffic and add an additional one if a higher degree of parallelism is required. You need to ensure that the number of session operators and predefined NetView tasks are identical.

Next, add a new monitor resource (MTR) that periodically requests exception information from this OMEGAMON session. Add the MTR by means of a *HasParent* relationship to the DB2 subsystem to be monitored. This ensures that the MTR will be activated when the DB2 subsystem is started, and deactivated when the DB2 subsystem is stopped. Also define the MTR via a *HasMonitor* relationship to the DB2 subsystem to ensure that the monitor's health status can be propagated to the application.

While the MTR is active, it uses the monitor command, INGMTRAP, to gather OMEGAMON exceptions that currently exist, based on the thresholds that are defined in the OMEGAMON for DB2 installation profile. INGMTRAP analyses all exceptions returned by OMEGAMON and filters out those exceptions that the MTR is interested in, in this example, LOGN. SA z/OS subsequently issues message ING080I to initiate exception processing.

Finally, also add a new rule to the automation table (via the SA z/OS policy) that executes a REXX exec to add a new log data set to the pool of primary active data sets whenever the LOGN exception is reported and the health status is `CRITICAL (6)`. The MTR's health status is considered CRITICAL if the number of available primary active logs is equal to 1. If the LOGN exception is reported again in the next monitor interval, a second rule in the automation table sets the MTR's health status to `FATAL (7)`, which triggers an application move because normal recovery

handling doesn't seem to work anymore. In addition, an alert is sent to the operator to inform him about this situation. If the LOGN exception is no longer reported, the MTR's health status will be set to `NORMAL (3)`.

The health status assigned to the MTR by means of the automation table is propagated to the DB2 application that owns this MTR. Thus, you can see at a glance whether the DB2 subsystem is okay or not.

Recommendations

You should consider the following recommendations when using OMEGAMON in combination with monitor resources:

- Avoid monitoring multiple exceptions using INGMTRAP. Note that there can be more than one exception that may trip and thus multiple ING080I messages may be generated. The monitor resource's health status, however, depends on the last ING080I message.
- Avoid setting different health statuses for the same exception that is monitored by different monitor resources using INGMTRAP. Note that only one automation table entry will be generated by SA z/OS to process message ING080I for such an exception.

In these cases, the use of INGOMX, invoked from an installation-written monitor command, to determine a combined health status from multiple exceptions or to determine an individual health status for each monitor resource, is preferred to using INGMTRAP.

Health Monitoring Based on OMEGAMON XE Situations

This section gives an overview of passive, event-based monitoring of OMEGAMON XE situations and describes how to set up the monitor resources using the customization dialogs.

Overview

Unlike the exception-based monitoring that SA z/OS uses for classic OMEGAMON monitors, the IBM Tivoli Monitoring infrastructure provides the means to react to situations whenever they occur. On the Tivoli Enterprise Portal (TEP), a user can specify what kind of automated response (reflex automation) should be triggered for each individual situation.

SA z/OS makes use of this capability by providing a simple command called INGSIT. The ITM administrator enters this command on the TEP with the Situation Editor dialog for those situations where SA z/OS health monitoring or health-based automation should take place. For more details about INGSIT refer to *IBM Tivoli System Automation for z/OS Programmer's Reference*.

The Take Action command is carried out on the agent, for example, OMEGAMON XE for z/OS, and not the Tivoli Enterprise Monitoring Server (TEMS) unless the TEMS is running on the same system. This is because it is possible that the hub TEMS may not reside on z/OS and so the command may not be delivered.

INGSIT triggers message ING150I that allows you to set the health status of individual monitor resources. It is then possible to issue commands, such as recovery or notification commands, to automatically fix the situation. You can specify what the health status is and what associated commands are issued in the customization dialog.

Defining the Monitor Resources

To set up the monitor resources:

1. Define one MTR for each OMEGAMON XE situation that you want to respond to.
2. In the MONITOR INFO policy item fill in the following fields:

   Monitored Object
   Enter the name of the OMEGAMON XE situation in uppercase with a prefix of ITM, for example, `ITM.MYSIT`

   Monitored Jobname
   Enter an optional job name to match this situation to a particular monitor resource.
3. Define codes for the message ID ING150I in the MESSAGE/USER DATA policy of the MTR to yield the commands that are to be issued and to map the severity to a valid health status.

Example Scenario: Consider the following scenario:

> The PAGEADD command is to be issued when an auxiliary storage shortage is detected, based on page data set utilization and page data sets that are not operational.

A situation called MyAuxShortage_Warn is defined by the installation that is true when both predefined situations OS390_Local_PageDS_PctFull_Warn and OS390_PageDSNotOperational_Warn are true.

As reflex automation, the following system command is issued on the managed system, that is, the system that produced the situations:

```
F NETV,INGSIT MyAuxShortage_Warn,warn
```

This command is issued from the Take Action dialog, as shown in Figure 6 on page 59.

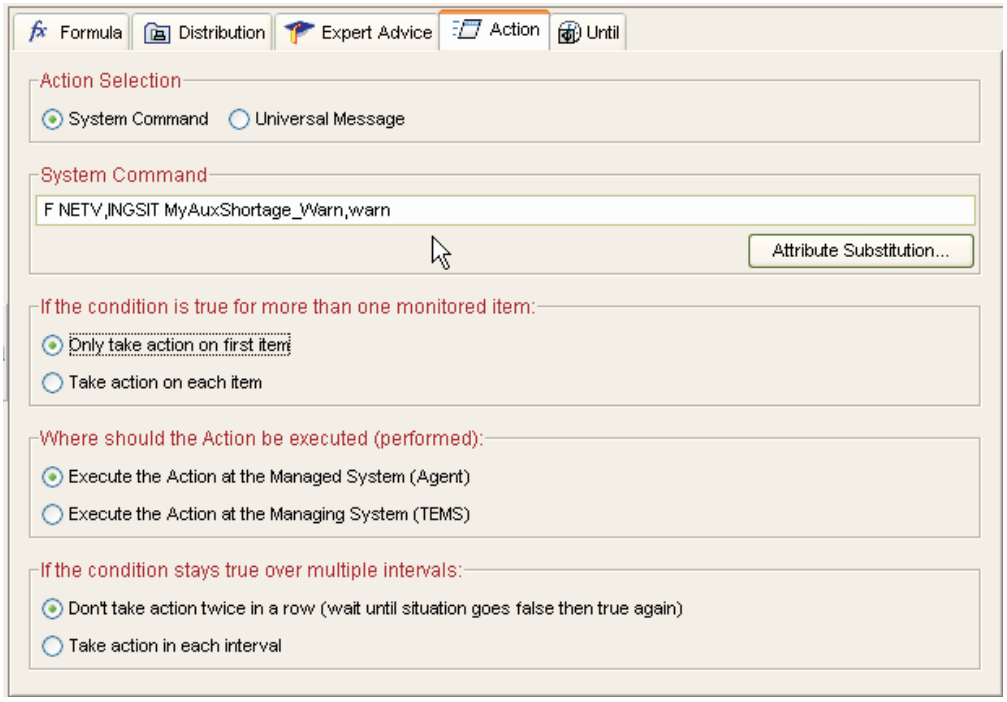


Figure 6. Take Action Dialog

INGSIT is called and produces an ING150I message, which contains the situation name that is mapped to the monitored object. Other optional information includes:

- The severity of the situation
- A job name that matches this situation to a particular monitor resource
- Other data that contains information related to the event

In this example, the situation, MyAuxShortage_Warn, and its severity, warn, are included.

Using the customization dialog, a monitor resource, for example, AUXSHORT, is created that specifies ITM.MYAUXSHORTAGE_WARN (in uppercase) as its monitored object.

ING150I is then specified in the MESSAGE/USER DATA policy item of the AUXSHORT monitor resource. In this example, the following code entry could be used to derive selection ADD and set the health status to MINOR:

```
Code 1          Code 2          Code 3          Value Returned
warn            *               *               ADD MINOR
```

In addition, one or more commands can be specified for ING150I for the selection that resulted from code match processing. In the example above, the PAGEADD command would be specified for selection ADD.

After executing all the commands that have been specified in the Command Processing panel for the selection, the health status that was mapped in the code processing is set (in this example, it was MINOR). Note that if no health status was specified in the code match table, it remains unchanged.

In a more sophisticated extension of this scenario, the situation, MyAuxShortage_Warn, as shown on the TEP is automatically acknowledged using SOAP services. To do this, a small request parameter XML-document must be created and sent to the TEMS SOAP server for processing. To acknowledge a situation, a CT_Acknowledge request must be issued as shown in the following example:

```
<CT_Acknowledge>
  <target>KEYAS:CMS</target>
  <name>MyAuxShortage_Warn</name>
  <source>KEYAPLEX:SYS1:MVSSYS</source>
  <data>System Automation is taking care of this</data>
</CT_Acknowledge>
```

The XML-document above references the TEMS that manages the situation (target), the situation itself (name), and the so-called monitoring agent (source) that is the source of this situation. With the data-element, you can pass any additional textual information to the person that is looking into this situation on the TEP.

As described in "OMEGAMON Interaction" on page 51, INGOMX is used to issue the SOAP request to the TEMS SOAP server. Once the situation has been acknowledged, it can be recognized as such on the TEP's situation event console or navigator flyover list.

Health Monitoring using CICSPlex SM

This section introduces the components of event-based CICS monitoring and describes how to set up the monitor resources using the customization dialogs.

Component Overview

Event-based CICS link and health monitoring is implemented using CICSPlex System Manager (CICSPlex SM) objects. Whenever an event is received from CICSPlex SM, message ING150I is issued.

INGCPSM is the event listener for CICSPlex SM. Because it is a long-running CLIST it needs to be run in a virtual operator station task (VOST). It scans the configuration on startup and listens for events. It then periodically checks whether the configuration has changed (that is, monitor resources have been added, deleted, or changed, etc.) or monitor resources are waiting for initial monitoring (that is, they have STATUS=ACTIVE and HEALTH=UNKNOWN).

Creating an Application to Manage the VOST

You can manage the VOST that executes INGCPSM using an application of type NONMVS:

- Start the VOST by using the INGVSTRT common routine as the start command of the APL, where its job name is used as the *attach_name* of the VOST.
- Stop the VOST using a sequence of INGVSTOP stop commands in the management APL.
- Monitor the status of the VOST using the INGVMON monitoring routine in the management APL.

For more details, see *IBM Tivoli System Automation for z/OS Programmer's Reference*.

Defining the Monitor Resources

To set up the monitor resources:

1. Define one MTR for each CPSM object (for example, each connection).

2. Fill in the **Monitored Object** field in the MONITOR INFO policy item according to the naming conventions, for example, CPSM.CICS1TOR.CONNECT.CON1
3. Leave the **Monitored Jobname** field empty.
4. Define codes for the message ID ING150I in the MESSAGE/USER DATA policy of the MTR to map the CPSM severities to valid health statuses, for example:

Code 1	Code 2	Code 3	Value Returned
VLS			* NORMAL
LS			* WARNING
LW			* WARNING
HW			* MINOR

Refer to the *CICS add-on policy for sample definitions to monitor the connection between two CICS resources.

Monitoring JES3 Components

The concept of a monitor resource is used to monitor the health of various JES3 components. SA z/OS provides two commands that support a strict separation of the monitoring part and the resulting recovery processing:

- AOFRJ3MN: used to monitor components in the JES3 environment, for example spool space.
- AOFRJ3RC: used to perform recovery actions against the monitored JES3 object.

The following example defines a spool space monitor:

1. Define a monitor resource with a “HasParent” relationship to the corresponding JES3 because it only makes sense to monitor the spool space when JES3 is active.
2. Activate and deactivate commands are not necessary for the spool monitor.
3. Use the AOFRJ3MN command as the monitor command and setup the monitoring interval as desired. In this example, spool usage of up to 60% is NORMAL, 61-70% WARNING, 71-80% MINOR, 81-90% CRITICAL and greater than 90% FATAL.

```
AOFRJ3MN  JES3_subys  SPOOLSHORT  60,70,80,90
```

4. Define the recovery action in the HEALTHSTATE policy, for example:

```
NORMAL  :  AOFR3RC   JES3_subsys  SPOOLSHORT  RESET
CRITICAL:  AOFRJ3RC  JES3_subsys  SPOOLSHORT  05
FATAL   :  AOFRJ3RC  JES3_subsys  SPOOLSHORT  01
```

   Issue one recovery command every minute. The commands are read from the SPOOLSHORT policy of the JES3 subsystem. When the spool usage goes down to 60% or less, the health status will go to NORMAL. This causes to invoke the AOFR3RC command but now with the RESET option - the RESET option stops recovery. It is recommended that you use JESOPER as the auto-operator for the recovery commands. Note, that the recovery commands for the SPOOLSHORT condition must be defined for the JES3 subsystem.
5. For the JES3 subsystem, define the necessary actions that should be performed for SPOOLSHORT in the Message/User data policy:

Pass	Automated Function	Command
1	JESOPER	MVS &SUBSCMDPFXF U,Q=HOLD,AGE=30D,N=ALL,C
2	JESOPER	MVS &SUBSCMDPFXF U,Q=HOLD,AGE=10D,N=ALL,C
3	JESOPER	MVS &SUBSCMDPFXF U,Q=HOLD,AGE=3D,N=ALL,C
10	JESOPER	MVS &SUBSCMDPFXF U,Q=HOLD,AGE=1D,N=ALL,C

This will purge all jobs from the hold queue that are older than 30 days in the first pass. On pass 2, all jobs older than 10 days are purged. On pass 3 all jobs older than 3 days are purged. Finally, after 10 times the pass interval (in our example 5 minutes), all jobs older than 1 day will be deleted if the recovery action is not reset meanwhile.

AOFRJ3MN Routine

Use this routine to monitor various objects in a JES3 environment. The following objects can be monitored:

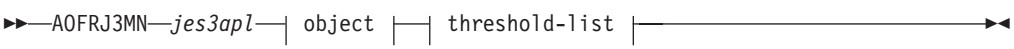
- MDS queues (Fetch queue, Verify queue, Wait volume queue, Error queue, Allocation queue, Breakdown queue, Unavailable queue, Restart queue, System select queue, System verify queue)
- Current setup depth
- Spool space

For each of the 10 JES3 MDS queues, thresholds may be set for each of the 4 health statuses (Warning, Minor, Critical and Fatal) indicating the number of jobs that particular queue may contain causing to set the corresponding health status. If, for example, the WARNING threshold for the Error queue is set to 5, if 5 or more jobs are pending on the MDS Error queue, the health status is set to Warning.

For the spool space the thresholds define the amount of used space that when exceeded causes to set the corresponding health status.

Whenever AOFRJ3MN is called, it issues the appropriate JES3 command (*I,Q,S for SPOOLSHORT and *I,S for the MDS queues) and parses the response. The value extracted from the message text is compared with the thresholds and then the return code is set to the corresponding health status. This simply sets the health status of the Monitor resource (MTR). No recovery action is taken by AOFRJ3MN routine. Use the HEALTHSTATE policy of the Monitor resource to define a recovery action for each health status, if necessary.

The syntax of the AOFRJ3MN routine is as follows:



object:

threshold-list:

jes3apl
Specifies the name of an APL of category JES3 for which this monitor works.

monitor
Specifies the JES3 object to be monitored:

MDSCOUNTQ	Current setup depth
MDSCOUNTF	Fetch queue
MDSCOUNTV	Verify queue
MDSCOUNTW	Wait volume queue
MDSCOUNTE	Error queue
MDSCOUNTA	Allocation queue
MDSCOUNTB	Breakdown queue
MDSCOUNTU	Unavailable queue
MDSCOUNTR	restart queue
MDSCOUNTSS	System select queue
MDSCOUNTSV	System verify queue
SPOOLSHORT	Spool

threshold-list
Specifies a list of four threshold values separated by commas:

warning	Set health status to WARNING if this value is exceeded
minor	Set health status to MINOR if this value is exceeded
critical	Set health status to CRITICAL if this value is exceeded
fatal	Set health status to FATAL if this value is exceeded

If *warning* is not exceeded the health status is set to NORMAL.

Note that for SPOOLSHORT the values are in percent but for the MDS queues they are absolute numbers. No value checking is done by AOFRJ3MN except for whole numbers.

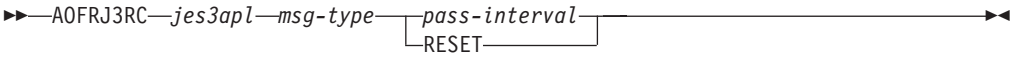
Note also that the thresholds are tested from FATAL to WARNING. So if you want to go directly from NORMAL to FATAL, you could specify `50,50,50,50`

AOFRJ3RC Routine

This routine performs the recovery action against a monitored object in a JES3 environment.

When AOFRJ3RC is called, it checks whether the system that it is running that holds the JES3 global processor. If not AOFRJ3RC terminates without any further action.

The syntax of the AOFRJ3RC routine is as follows:



jes3apl Specifies the name of an APL of category JES3.

msg-type
Specifies the message type within the given JES3 APL that the recovery commands are to be read from:

pass-interval
Specifies the time interval that AOFRJ3RC should wait before executing the next pass. The format is in NetView notation (mm, hh:mm, hh:mm:ss or :ss).

RESET
If RESET is specified AOFRJ3RC stops the recovery.

AOFRJ3RC looks into the MESSAGE/USER DATA policy definition of the specified JES3 APL. It issues the command that is defined for PASS1 of the given message type. As long as there are commands in higher passes it sets up a NetView timer that re-calls AOFRJ3RC after the given pass interval. Whenever AOFRJ3RC is executed the command that is defined for the next pass is issued as long as one exists.

If RESET is specified instead of a pass interval any pending timer is killed and processing stops.

The return code is always zero.

Note: AOFRJ3RC issues the recovery commands in a *fire-and-forget* manner. It does not check whether the recovery action has the desired result. This is done by the monitor. After one or more monitor intervals the health status will change to a less severe one if the recovery shows an effect. If you want to stop recovery actions when the health status returns to NORMAL, for example, you have to code a HEALTHSTATE command that calls AOFRJ3RC with RESET.

IMS Component Monitoring

For IMS automation, SA z/OS enables the monitoring of online log data sets (OLDS) and recovery control data sets (RECON) of IMS control regions, and allows the status checking of the VTAM Application Control Blocks (ACB) and the enablement of logons.

The monitor routines that are provided for this and the necessary definitions to enable the monitoring functions are described in the *IBM Tivoli System Automation for z/OS IMS Automation Programmer's Reference and Operator's Guide*.

Chapter 5. Alert-Based Notification

SA z/OS enables you to use the notification feature of System Automation for Integrated Operations Management (SA IOM) to alert subject matter experts if SA z/OS encounters problems that require manual intervention. Notification can be via SMS, e-mail, pager or telephone and is especially useful in unattended situations.

Overview

The interface between SA z/OS and SA IOM allows alerts to be sent to operators or systems programmers for predefined situations. You can also customize when to issue alerts, if desired, using the customization dialog and the INGALERT programming interface. Alerts can be issued only for applications (APL), monitor resources (MTR), and application groups (APG).

An alert is information that is collected and sent by an automation agent to SA IOM for notification processing. The information that is passed to SA IOM consists of:

- The escalation ID that indicates the rules determining how the alert should be processed
- The priority of the alert
- Text to be sent to the alerted person or group

There are several predefined *alert points* that trigger alerts whenever a generic routine encounters a problem situation, such as a resource becoming degraded or when an IMS shutdown check fails.

Alerting can be enabled or disabled globally using the INGCNTL command. Alerts are only produced for a resource if the Inform List field in the resource's policy contains IOM.

Communication Flow

Figure 7 on page 68 outlines the communication between the automation manager and the automation agents.

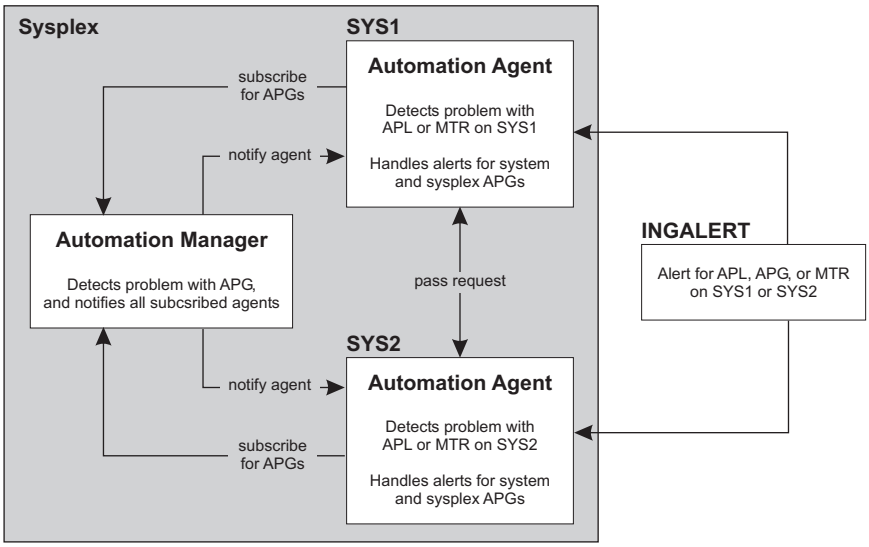


Figure 7. Alert Communication Flow

The automation agents on the systems in the sysplex subscribe to the automation manager to be alerted about problems with system or sysplex application groups (APGs). This is because the automation manager cannot itself send alerts to the SA IOM server. Whenever the automation manager detects a problem with an APG it sends an alert to the subscribed automation agents (one in case of a system APG, and all in case of a sysplex APG). Any alerts for sysplex APGs are handled by only one automation agent in the sysplex.

The automation agents can also receive alerts for applications, application groups, or monitor resources via the INGALERT command. If the affected resource is managed by a different automation agent, the request is passed on. The automation agent that manages the resource sends the alert to SA IOM. If, for whatever reason, this automation agent cannot send the alert, it passes on the request to the next automation agent in the sysplex. This can happen several times until the alert is successfully sent or no more automation agents are available.

For each alert, the automation agent connects to SA IOM, sends the alert and then disconnects. The automation agent does not maintain a permanent connection to the SA IOM server.

Enabling Alerting

By default alerting is not enabled. To activate it you must perform setup actions in both SA z/OS and SA IOM.

Setup in SA z/OS

You can turn alerting on or off at three different levels in SA z/OS:

- The system level, via the INGCNTL command. Turning off alerting means that no alerts are detected or accepted by the system. Alerting must be turned on explicitly.
- The resource level, via the Inform List policy field. Turning off alerting means that no alerts are detected or accepted for the resource. IOM must be explicitly specified (or inherited from the defaults) to activate alerting.

- The alert ID level, via codes for the INGALERT message ID of the resource or MVC entry.

INGCNTL Command

By default alerting is not enabled. You have to issue the INGCNTL command to enable it and set the connection properties to the SA IOM server. This can be done as follows:

- In the NetView style sheet using auxiliary commands:

```
*********************************
* Auxilliary commands
*********************************
* Enable Alerting and set connection properties
auxInitCmd.A = INGCNTL SET ALERTMODE=ON ALERTHOST=saiom:1040:SAALERT
```

- From the exit AOFEXDEF that is supplied with SA z/OS:

```
'INGCNTL SET ALERTMODE=ON ALERTHOST=saiom:1040:SAALERT'
```

See *IBM Tivoli System Automation for z/OS Programmer's Reference* for more information about the INGCNTL command.

Inform List

You can specify IOM in the Inform List field to explicitly enable alerting for specific resources or classes of resources, as shown in Table 4.

Table 4. Inform List Policy Items

Policy Object	Policy Item
Application Group (APG)	APPLGROUP INFO *
Application (APL)	APPLICATION INFO *
Monitor Resource (MTR)	MONITOR INFO *
MVSCOMP Defaults (MDF)	MVSESA INFO *
System Defaults (SDF)	AUTOMATION OPTIONS
Sysplex Defaults (XDF)	RESOURCE INFO
* Leaving the Inform List field blank allows the policy object to inherit the value specified in the system defaults definition.	

Code Processing

Code processing with the INGALERT message ID allows you to define a priority and escalation ID to be passed to SA IOM. You can use this for resources of type APL, APG MTR, and MVC. If no matches are found for the APL, APG or MTR resources, the INGALERT codes are checked for the corresponding MVC entry on the system where the resource resides.

Enter the following in the Code Processing panel for the INGALERT message ID:

Code 1

The alert ID that identifies the type of alert. SA z/OS provides the following set of built-in alert points:

Alert ID	Description	For Resource Type
AS_PROBLEM	Shutdown check for feature failed	APL
CMD_FAILED	Return code checking is on and the command ended with RC≠0	APL, MTR
IMPORTANT_WTOR	Important WTOR triggered OUTREP	APL
OS_DEGRADED	Resource has become degraded	APL

Alert ID	Description	For Resource Type
OS_PROBLEM	ZOMBIE, BROKEN or shutdown outside SA z/OS and restart not allowed	APL
START_FAILED	Start command failed	APL
START_PENDING	Up message not received within timeout interval	APL
STOP_PENDING	Ran out of stop commands	APL
CS_PROBLEM	Compound status PROBLEM has been set	APG

You can also use any user-defined alert ID. Simply specify it in the corresponding code entry and call INGALERT with this ID. Wildcards are supported.

Code 2

The job name that alerting should be done for. Wildcards are supported. You can use Code2 only for MVC and APL definitions. This allows you to set alerting for several APLs at once by using APL classes.

Code 3

Leave this field empty.

Value Returned

This can be either `IGNORE` or the two tokens *`priority escalation_id`* where:

- *priority* is the priority of the alert (0–999).
- *escalation_id* is the ID that is used in SA IOM to define the rules that determine how the alert should be processed.

Consider the example in Figure 8.

```
Code 1          Code 2          Code 3          Value Returned
AS_PROBLEM      *                               10 StandBy1
*               TST*                            IGNORE
*               *                               100 StandBy2
```

Figure 8. Code Processing Example for the INGALERT Message ID

This example is defined on a class. It results in the following behavior:

- All alerts with the alert ID AS_PROBLEM are sent to the escalation ID StandBy1 with a priority of 10.
- Any alerts for jobs starting with TST (for example, test resources) are ignored.
- The remaining alerts are sent to the escalation ID StandBy2 with a priority of 100.

Setup in SA IOM

SA z/OS uses the peer-to-peer protocol of SA IOM. You need to define the peer connections in SA IOM as follows:

1. Click the **Peers** tab in Server Configuration Properties notebook.
2. Specify a port to use for peer connections, for example, 1040. This port must also be specified with the ALERTHOST parameter in the INGCNTL command.

3. Right-click in the IP addresses pane and specify the following for each z/OS machine that is running INGALERT:
   - The IP address or host name, for example, 10.192.0.1 or SAIOMSRVR
   - A description for the peer connection
4. Click **OK**.

INGALERT Command

You can use the INGALERT command to inject alerts into a system. This can be from either the NetView automation table, a REXX script or the command line.

You can specify the following parameters:

- A resource name , the text MVSESA, or a job or subsystem name.
- The alert ID, for example, CS_PROBLEM, CMD_FAILED, etc.
- A message ID that identifies the message text or a text string that is passed to SA IOM.

For example, the following can be used from the command line or a CLIST:

```
INGALERT MYGRP/APG/SYS1 ID=MYALERT TEXT=(MYGRP HAS A PROBLEM)
```

It uses the alert ID, MYALERT, to define the escalation ID and priority of the alert, and the specified alert text.

The following can be used from the NetView automation table to send an alert whenever message ABC123I is issued:

```
IF MSGID='ABC123I'
THEN
EXEC(CMD('INGALERT'));
```

It uses the alert ID ABC123I and the complete message text of ABC123I as the alert text.

See *IBM Tivoli System Automation for z/OS Programmer's Reference* for more information about the INGALERT utility.

Chapter 6. Availability and Recovery Time Reporting

SA z/OS introduces support to assist you in billing users or reporting reliability of your critical applications or the software that those applications are dependent on. For example, you might want to charge accurately based on the amount of time required to run an application. This is of importance for non-MVS resources, such as USS applications, or monitoring resources that might run in the NetView address space.

Overview

SA z/OS collects and records job-related information, and writes System Management Facility (SMF) records at specific events in the lifetime of a resource. This resource can be:

- A subsystem (APL)
- An application group (APG) that is hosted by the local system as well as sysplex application groups
- A monitor resource (MTR)

The INGPUSMF batch utility produces a report file that you can import into a spreadsheet.

You can control whether a record is written for a resource by entering the value SMF in the Inform List field in the resource's information policy item.

Resource Lifecycle

Figure 9 shows the events in the lifetime of an application when SA z/OS records information.

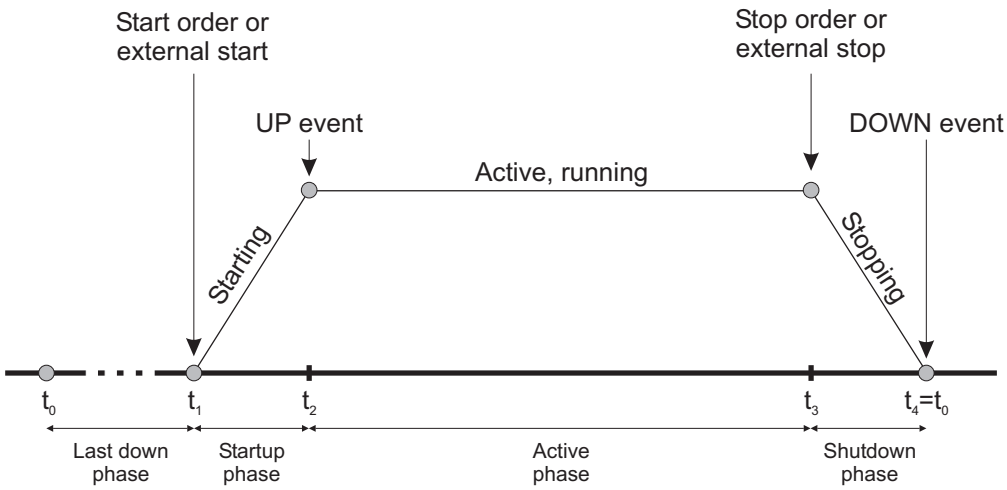


Figure 9. Events in the Lifecycle of an Application

These events are:

- Start order received from the automation manager (t_1)
- UP signal received (t_2)

- Stop order received from the automation manager (t_3)
- DOWN signal received (t_4=t_0)

By examining these records you can establish the following information for a given time period:

- Application up time and downtime
- Application startup and shutdown time
- The number of scheduled stoppages and the approximate amount of scheduled downtime
- The number of unscheduled stoppages and the approximate amount of unscheduled downtime

To make using SA z/OS SMF records easier, the following periods are automatically calculated and stored (in units of seconds) in the SMF record:

- The startup time (t_2–t_1)
- The shutdown time (t_4–t_3)
- The time the application was active (t_3–t_2)
- The last down time (t_1–t_4)

You therefore have a precise view of the lifecycle of the application.

Layout of the SMF Record

Table 5 provides details of the data that is stored in the SMF record.

Table 5. Layout of the SMF Record

Offset	Length	Format	Description
00	2	Binary	Record length. This field and the next (a total of 4 bytes) form the record descriptor word (RDW).
02	2	Binary	Segment descriptor. This is zero.
04	1	Binary	System Indicator Bit: 0 Reserved 1 Subtypes used
05	1	Binary	SMF Record Type. This is 114.
06	4	Binary	The time, since midnight, that the record was moved into the SMF buffer (in hundredths of a second).
10	4	Packed	The date when the record was moved into the SMF buffer, in the form 0*cyyddd*F.
14	4	EBCDIC	System Identification (from the SID parameter).
18	2	Binary	Record subtype: 1 Automation tracking record
20	2	Binary	Record version.
22	2	—	Reserved.
24	4	Binary	Offset to product section from start of record, including the record descriptor word (RDW).
28	2	Binary	Length of product section.
30	2	Binary	Number of product sections. This is always 1.
32	4	Binary	Offset to resource section from start of record, including the record descriptor word (RDW).
36	2	Binary	Length of resource event section.

Table 5. Layout of the SMF Record (continued)

Offset	Length	Format	Description
38	2	Binary	Number of resource event sections. This is always 1.
40	8	—	Reserved.
Product Section			
00	16	EBCDIC	Product name, for example SA z/OS V3R2M0.
16	8	EBCDIC	Name of the SYSPLEX.
24	8	EBCDIC	Domain identifier.
32	8	EBCDIC	MVS System name.
40	8	EBCDIC	XCF group name.
Automation Section			
00	24	EBCDIC	Resource name (in automation manager notation).
24	8	EBCDIC	Job name (optional).
32	2	Binary	Event type: X'0001' Starting X'0002' Active X'0003' Stopping X'0004' Inactive X'0005' Degraded
34	2	—	Reserved.
36	12	EBCDIC	Automation agent status (optional).
48	12	EBCDIC	Start type.
60	12	EBCDIC	Stop type.
72	5	EBCDIC	Termination type (abend code). Optional.
77	3	—	Reserved.
80	4	Binary	Total startup time in seconds.
84	4	Binary	Elapsed time in seconds that the resource was active.
88	4	Binary	Total shutdown time in seconds.
92	4	Binary	Last down time of resource in seconds.

The INGPUSMF Utility

You can use the INGPUSMF utility to analyze SMF records and produce a data set that can be imported into a spreadsheet program. The data set contains the type 114 records that SA z/OS produces in a format that can easily be imported. By default, the fields are semicolon delimited.

Output

The first record in the data set is a title record that describes each column. The remaining records are the data records. One data record is written for each type 114 SMF record.

Table 6 describes the format of each record.

Table 6. Format of INGPUSMF Utility Data Set Records

Column	Description
1	SMF system ID

Table 6. Format of INGPUSMF Utility Data Set Records (continued)

Column	Description
2	Date when SMF record was written, in YYYYMMDD format
3	Time when SMF record was written, in hhmmss format
4	SA z/OS product name, including release level
5	Name of sysplex
6	System name
7	NetView domain ID
8	XCF group name
9	Resource name in automation manager notation
10	Job name, if present
11	Event
12	Automation agent status
13	Startup time in seconds
14	Active time in seconds
15	Shutdown time in seconds
16	Down time in seconds
17	Start type
18	Stop type
19	Termination (abend) code

The INGPUSMF Utility JCL

The INGPUSMF utility runs as a batch job. See INGEUSMF for a sample. The meaning of the DD statements is as follows:

STEPLIB
The load library that contains the INGPUSMF utility. The utility resides in the SINGMOD1 library.

REPORT
The output data set that contains the spreadsheet import data set in a semicolon-delimited format. The record size is 255 bytes.

SYSPRINT
Contains information that is written by the utility.

HSATRACE
Is used for debugging purposes only. If present, the INGPUSMF utility writes trace entries to record the process flow.

SMFDATA
Contains the SMF records. The record format is: Variable, blocked, spanned.

USRPARMS
Contains user options, such as filter criteria or a specific separator character.

User Options

You can specify various options in the USRPARMS data set that control the processing of the utility. You must specify each option in a separate record. The

option are defined as keyword=value pairs. If you specify an option several times, the last occurrence is used. The keyword must start in column 1 of the record. No blanks are allowed in front of or after the equal sign (=). A asterisk (*) is considered to be a comment.

The following options are supported:

SEPCHAR=*char*
: Defines the separator character to be used to separate the columns. The default is a semicolon (;) if omitted

SYSID
: Defines the SMF system ID used as a filter. Only SMF records that are generated by that system are taken. The value can be 1–4 characters.

FROM=*date*
: The starting date used as a filter. The format is YYYYMMDD. All SMF records written on the specified date or later are taken.

TO=*date*
: The ending date used as a filter. The format is YYYYMMDD All SMF records that are written no later than the specified date are taken.

RESOURCE=
: Defines the resources in automation manager notation used as a filter. You can specify up to 10 resource names. The name can be a wildcard, such as *abc, abc* or *abc*.

Return Codes

The following return codes are set by the utility:

0 Normal completion.

8 Invalid option detected in the USRPARMS data set.

12 REPORT data set is not accessible.

16 A severe error occurred, for example, an open error for the SMFDATA data set, or writing a record to the REPORT file.

Chapter 7. How to Automate Your Resources

This chapter contains information on how to customize your SA z/OS installation by programming various routines and procedures. It describes various ways that you can adapt your installation to your requirements.

Using Automation Flags

SA z/OS extended automation flags (automation flags for minor resources) give you the ability to control the automation for individual messages and status changes. You cannot use extended automation flags to stop a status change from occurring, but you can use them to stop commands or replies being issued in response to a change to a particular status.

You can define messages and status information as minor resources with a major resource that is either:

- The application that issued the message or changed status
- MVSESA, if the message or status change is not associated with an application.

To define messages or status information as minor resources, use the customization dialog to edit the *Minor Resources* policy item of the appropriate Application policy object or the *MVS Component* policy object. See *IBM Tivoli System Automation for z/OS Defining Automation Policy* for more information.

When an application is about to change to a new status, the status change routines (ACTIVMSG, HALTMSG and TERMMSG) check whether the new status has been defined as a minor resource for the application before they issue any commands associated with the status change. See *IBM Tivoli System Automation for z/OS Programmer's Reference* for more information about SA z/OS routines.

The command and reply routine (ISSUEACT) checks to see whether the message ID of the message that triggered them is defined as a minor resource under the associated application (or under MVSESA for a system message).

Note: Calling ISSUEACT with AUTOTYP=NOCHECK disables this checking, but because it causes a number of incongruities, this is not recommended.

By default a minor resource inherits the automation flag settings of its major resource. You can use the customization dialog or INGAUTO to set specific flags for minor resources. You can see the current automation flag settings for both major and minor resources on the DISPFLGS panel.

Example

When TSO issues message IKT001D and this is trapped by an automation table statement that runs ISSUEACT AUTOTYP=START, the following actions are taken:

1. The TSO Start flag will be checked
2. If either the TSO Start flag is turned on or minor resource checking is enabled, the TSO.IKT001D Start flag is checked.
3. If the TSO.IKT001D Start flag is turned on, ISSUEACT issues any replies appropriate to the message.

4. If the TSO.IKT001D Start flag is turned off (even though the TSO Start flag may be turned on), SA z/OS does not attempt to reply to the message.

When SA z/OS Checks Automation Flags

This section describes how SA z/OS uses automation flags. It provides background information to help you customize SA z/OS-provided automation and to help you write your own automation procedures.

Figure 10 shows how specific automation flags cover the phases in the lifetime of an application.

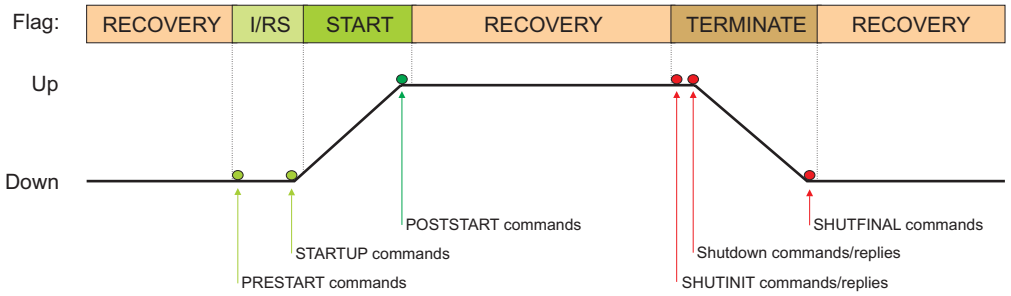


Figure 10. Automation Flag Span of Control

Table 7 summarizes how SA z/OS typically uses automation flags. SA z/OS provides a common routine, AOCQRY, to perform automation flag checking. See *IBM Tivoli System Automation for z/OS Programmer's Reference* for a description of AOCQRY.

Table 7. Automation Flags: Typical Uses in SA z/OS

Automation Flag	Typical Use In SA z/OS
Automation	Checked before any other automation flag to determine whether overall automation for the resource is on or off. If it is off, none of the other flags is checked.
Initstart	Checked after IPL when the application has a true DOWN status. If this is on, SA z/OS will start the resource, provided its goal is to be available.
Recovery	Checked to determine whether to proceed with performing recovery actions other than restarting a resource.
Restart	Checked to determine whether the resource is eligible for restart.
Terminate	Checked for all shutdown commands and for message automation during the shutdown phase.
Start	Checked for message automation after the STARTUP commands have been issued and for POSTSTART commands.

Automation Flag Exits

SA z/OS will invoke an exit only if it needs to in order to evaluate a flag. For example, if an exit is specified on a subsystem restart flag but the global SUBSYSTEM Automation flag is off, SA z/OS does not invoke the exit when it checks the restart flag because the setting for the subsystem Automation flag (inherited from the SUBSYSTEM Automation flag) is off.

If the situation is reversed (exit for the subsystem Automation flag and the SUBSYSTEM Restart flag is off) the exit would also not be invoked. See "Flag Exits" on page 159 for more information on automation flag exits.

Do not rely on SA z/OS to invoke an exit every time a flag is checked. You can only rely on SA z/OS to invoke an exit before it concludes that a flag is turned on.

The Automation Manager Global Automation Flag

Using the INGLIST or the INGSET command (see *IBM Tivoli System Automation for z/OS User's Guide* or *IBM Tivoli System Automation for z/OS Operator's Commands*) you can set an automation flag for the individual resources, which is checked by the automation manager before it sends any order to the automation agent to start or stop the specific resource.

The purpose of this flag is to prevent (if flag is set to NO) or enable (YES) the starting or stopping of resources. This can be done for resources that reside on systems that are currently inactive, for example, to prevent the startup of the resource at IPL time of the system.

Chapter 8. How to Automate Processor Operations-Controlled Resources

This chapter contains information on how to customize your SA z/OS installation to enable the automation of messages coming from target systems that are controlled by processor operations. These target systems or resources are referred to as ***processor operations resources*** in the following.

Notes:

1. VM guest systems are treated the same as any other target systems that are controlled by ProcOps (see *IBM Tivoli System Automation for z/OS Operator's Commands* for details).
2. PSMs are "virtual" hardware and therefore not all Target hardware commands apply (see *IBM Tivoli System Automation for z/OS Operator's Commands* for details).

With the method described in this chapter, you can use SA z/OS system operations to react on these messages. This information is contained in "Automating Processor Operations Resources of z/OS Target Systems Using Proxy Definitions," which introduces the general process how to achieve such message automation.

Automating Processor Operations Resources of z/OS Target Systems Using Proxy Definitions

SA z/OS processor operations can be used to automate messages that cannot be automated on the target systems themselves. Typically these messages include those appearing at IPL time.

In a sysplex environment there are additional messages (XCF WTORs) being displayed at IPL time when joining the sysplex and at shutdown time when a system is leaving a sysplex. These WTOR messages cannot be automated yet because SA z/OS system operations is not active at that time.

With the XCF message automation framework described in this chapter, you have a method of exploiting your own XCF message automation.

Note: There are XCF WTOR messages which are automatable by Sysplex Failure Management (SFM). In these cases, to avoid conflicting automation, it is not recommended that you automate these messages by SA z/OS.

Concept

You can use the SA z/OS standard interface and routines to handle system external messages in almost the same way as system internally generated messages. This applies to the way of defining message automation in the customization dialog as well as to the means available for controlling message automation at automation time.

To exploit the system operations mechanism for message automation, a *proxy resource* representing the processor operations resources must be generated in the customization dialog as entry type Application (APL).

There is a one-to-one relation between a proxy and a processor operations resource (target system). How to implement this relation in the customization dialog is described in the following subsections.

Messages that are generated on external systems, where no SA z/OS is active or not yet active, can also be automated. The resources generating these messages are called *processor operations resources*. They are defined in the customization dialog as entry type System (SYS).

Customizing Automation for Proxy Resources

It is assumed that you have already used the customization dialog to define processor operations target systems and made these systems accessible to the processor operations focal point via the Processor Control file (see also *IBM Tivoli System Automation for z/OS Defining Automation Policy*). So for every processor operations target system that has been defined on the processor operations focal point, you should define a proxy resource. You do this by defining the proxy resource as entry type Application (APL) in the customization dialog.

Note: If you want to define many proxy resource applications, you can use the application class concept as described in *IBM Tivoli System Automation for z/OS Defining Automation Policy*).

Defining the proxy resource as an Application (APL) has another advantage: The system is then visible in the INGLIST panel and it can be managed and monitored like an application resource. SA z/OS users will be able to not only use message automation for target system messages, they can also issue start and stop commands to IPL and shut down systems. These commands can be defined like any start and stop command for an application. Unlike application resources, target systems are managed by processor operations commands (for example, ISQCCMD *target_system_name* ACTIVATE FORCE(NO) or ISQSEND *target_system_name* OC vary xcf,*target_system_name*,off,retain=yes). Processor operations commands allow you to send MVS commands to target systems as well as hardware commands to the processor (support element).

The rules that you need to obey when defining the proxy resource are as follows:

1. You need to define (or have defined) the processor operations target systems that you want to automate. For those systems, the following rule applies:
   MVS SYSNAME = ProcOps name

   The **MVS SYSNAME** must be identical with the **ProcOps name**.

   If this is not the case, you need to change it subsequently.
2. **Job Name = ProcOps name**

   The **Job Name** of the application for the proxy resource must match the processor operations target system's name as defined when creating this system in the customization dialog.
3. **Job Type = NONMVS**

   The **Job Type** for the proxy application must be NONMVS.
4. The **Monitor Routine** for the proxy application must be ISQMTSYS.
5. **Sysname = MVS SYSNAME**

   The **Sysname** for the proxy application must match the **MVS SYSNAME** defined for the processor operations target system. This definition is used for resource monitoring.

6. If you want to inhibit operators from performing a startup or shutdown for a target system resource using the INGREQ command, **External Startup** and **External Shutdown** must be set to 'ALWAYS'.
7. If you do not want the proxy resource to be automatically started, you should set the **Restart after IPL** option to NO.
8. Because you can only automate applications by linking them to systems via an application group, you need to define an application group for the proxy applications. Do not merge the proxy applications with other applications into this application group because destructive requests applied to a merged application group would also affect the proxy resources contained in that group.

   You may choose PASSIVE behavior to not forward requests against the application group to each member. This will prevent you from unintentionally sending requests to processor operations target systems represented by their proxies.
9. In the Message Processing panel for the proxy application define the messages to be automated in the **Message ID** column . Do not specify message ID ISQ900I, as this message is used as a carrier for the original target system message.

   Enter `cmd` in the **Action** column to specify the command to be processed if the defined message occurs.
10. If the message to be automated is a WTOR, then the variable &EHKVAR1 will contain the reply ID. This variable may then be used as a parameter to the ISQSEND command:

    ```
    ISQSEND &SUBSJOB OC R &EHKVAR1,COUPLE=00
    ```

Startup and Shutdown Considerations

Processor operations commands must be used to start or stop processor operations resources, for example:

- Start example:

  ```
  ISQCCMD &SUBSJOB LOAD FORCE(NO)
  ```

- Stop example:

  ```
  Pass 1 ISQSEND &SUBSJOB OC Z EOD
  ```

  ```
  Pass 2 ISQSEND &SUBSJOB OC VARY XCF,&SUBSAPPL,OFF,RETAIN=YES
  ```

> **Note:**
> If the delay time between sending the commands in pass 1 and pass 2 is not appropriate, you can define a resource-specific Shut Delay in the Application Automation Definition panel.

For more details about processor operations commands refer to *IBM Tivoli System Automation for z/OS Operator's Commands*.

Preparing Message Automation

The interaction with target systems is based on the SA z/OS processor operations component. Therefore the installation and customization of this component must be complete at this point.

Operating System messages from processor operations target systems receiving at the focal point will be transferred to ISQ900I messages.

ISQ901I is not relevant. It is used to inform interested operators about target system messages. It is not used for automation purposes.

MSCOPE() parameter in CONSOLxx member

MSCOPE allows you to specify those systems in the sysplex from which this console is to receive messages not explicitly routed to this console. An asterisk (*) indicates the system on which this CONSOLE statement is defined. Because the default is *ALL, indicating that unsolicited messages from all systems in the sysplex are to be received by this console, this parameter must be set to '*' for correct automation by SA z/OS processor operations.

Automating Linux Console Messages

The Linux Console Connection to NetView

When a Linux target system IPLs, its boot messages are displayed on the Console Integration facility (CI) of the zSeries or 390-CMOS processor Support Element (SE). For SA z/OS processor operations, CI is the only supported interface to communicate with the Linux operating system. The communication between the processor operations focal point and CI is based on the NetView RUNCMD and the Support Element's Operator Command Facility (OCF), an SNA application. In SA z/OS processor operations, this connection path is referred to as a NetView Connection (NVC).

Linux Console Automation with Mixed Case Character Data

Unlike operating systems which translate console command input into uppercase characters, Linux is case sensitive. The NetView automation table syntax allows the use of mixed case characters in compare arguments of an IF statement. When an automation command is to be scheduled as a result of such a comparison, any message token arguments passed, are not translated into uppercase by NetView. Make sure that your automation routine does not do an uppercase translation of parameters passed. For example, in REXX use the statement 'PARSE ARG P1 P2' instead of 'ARG P1 P2', which implicitly performs a translation into uppercase. If a Linux message invokes your automation code and the message information is retrieved using NetView's GETMLINE function, no uppercase translation will occur. In order to send mixed case command data to the Linux console consider the following REXX statement:

```
Address Netvasis 'ISQsend MYlinux Oc whoami'
```

The addressed REXX command environment 'Netvasis' passes the command string without doing an uppercase translation. The ISQSEND command internally translates its destination parameters into 'MYLINUX' and 'OC' but leaves command 'whoami' as is.

Security Considerations

After Linux system initialization, usually a LOGIN prompt message is displayed allowing users defined to the system to login. The ISQSEND command interface does not suppress any password data from being displayed. You may use the NetView LOG suppression character to avoid the password information to be visible in the NetView log. In SNA/VTAM traces or Support Element log files, such password data can be viewed in text form.

Restrictions and Limitations

The following Linux systems are supported:

- Linux systems running in an LPAR of a zSeries or 390-CMOS processor hardware
- Linux systems running on a zSeries or 390-CMOS processor hardware, configured in Basic mode
- Linux systems running as VM guest machines under z/VM Version 4.3 or higher

Linux systems running under a VM, which itself runs as a VM guest, are not supported.

In the command shell environments of a Linux console it is possible to pass control keys as character strings instead of pressing the keyboard control key combination to perform functions like Control-C. The current Linux support of SA z/OS processor operations has not been tested using this Linux capability. Any Linux program or command script that requires a user interaction with control keys should not be invoked using the SA z/OS processor operations ISQSEND interface.

How to Add a Processor Operations Message to Automation

Use the NetView automation table (AT) and the SA z/OS command set to implement console automation. You can automate the routine functions that an operator performs when a particular message is generated. For more information see *IBM Tivoli System Automation for z/OS Defining Automation Policy*, SC33–7039.

Messages Issued by a Processor Operations Target System

When a target system issues a message, the message is forwarded to the processor operations focal point system. The focal point system repackages the message within an SA z/OS ISQ900I message, an ISQ901I message, or both, and routes the message to the appropriate task:

- ISQ900I messages are routed to SA z/OS processor operations autotasks. If you want automation that you write to receive ISQ900I messages, use the ISQEXEC command to run the automation in a target control task. For information about using the ISQEXEC command, see section *Sending an Automation Routine to a Target Control Task* in "Issuing Other OCF Commands" on page 10. Your NetView automation table entries for SA z/OS should acknowledge the ISQ900I identifier for all target system messages forwarded to the processor operations focal point system. You can specify your ISQ900I automation table entries to be target system specific, however, this is not recommended.
- ISQ901I messages are routed to all logged-on operators identified as interested operators by the ISQXMON command or marked as such in the customization dialog.

For information about the ISQEXEC and ISQXMON commands, see *IBM Tivoli System Automation for z/OS Operator's Commands*.

A message forwarded from a NetView connection or an SNMP connection consists of the following:

- ISQ900I or ISQ901I message identifier
- Name of target system where the message originated
- Console designator form describing where the message originated
- Message identifier and text of the original message from the target system

For example, if a NetView connection forwards the message IEA101A SPECIFY SYSTEM PARAMETERS from the operating system to the focal point system, SA z/OS creates one or both of the following SA z/OS messages:

```
ISQ900I target-system-name OC IEA101A SPECIFY SYSTEM PARAMETERS
ISQ901I target-system-name OC IEA101A SPECIFY SYSTEM PARAMETERS
```

This message format applies to all processor operations target system messages. It is independent of the target system resource that generated the original message.

The processor operations target system message is sent in the same format as it would be displayed on the processor Support Element (SE) or Hardware Management Console (HMC).

Specifics of VM second level systems:

Messages from guest machine operating system appear in the following format:

```
ISQ900I psm-name.guest-name OC IEA101A SPECIFY SYSTEM PARAMETERS
```

Messages from CP on the virtual machine appear in the following format:

```
ISQ900I psm-name.guest.name OC HCPGSP2627I The virtual machine is
        placed in CP mode due to a SIGP initial CPU reset from CPU 00.
```

Messages from the PSM itself appear in the following format:

```
ISQ700I psm-name SC ISQCS0314E Message Handler has failed.
```

Note:

Make sure your consoles issue messages in the format that you expect and write your NetView automation table entries accordingly.

Sample NetView Automation Table Statements

The following message response example presents a request for system parameters when the message ID string contains 'IEA101A':

```
IF      TEXT = . 'IEA101A SPECIFY SYSTEM PARAMETERS'
      & MSGID = 'ISQ900I' .
THEN    EXEC( CMD('ISQI101 ' ) ROUTE ( ONE *  ))
        DISPLAY(N) NETLOG(Y);
```

This NetView automation table statement initiates the ISQI101 routine when the message condition is true.

Note: Text within messages may be in mixed case. Be sure your coding accounts for mixed case text.

Message ISQ211I

Some SA z/OS commands attempt to lock and unlock ports. Where an operator owns the lock for a port, the SA z/OS unlock command, ISQXUNL, returns RC=12 associated with message ISQ211I `Unable to unlock` *`target name console`*.

In such a case, you have the choice of either using the ISQOVRD command to force an unlock or you may end your automation with a message. Thereafter, you can view your NetView log to find out the reason for the lock of the port.

Your automation may encounter this message ISQ211I frequently. Attempting to unlock a locked port is not an error condition; however, it may be a sign that the calling command did not succeed. Schedule your automation from messages that indicate positively that a command did not run, not from the ISQ211I message.

Processor Operations Command Messages

Some SA z/OS commands run on the target system. The message returned from these commands indicates only that the support element was told to schedule the operation. Consequently, the operation at the target system may not complete even though the SA z/OS message indicates a successful completion.

SA z/OS acknowledges only that the command was successfully forwarded to the support element. An unsuccessful operation at the target system generates an unsolicited message that the support element forwards to the focal point system in an ISQ900I message. Schedule your automation from the message that positively indicates that a target system operation did or did not complete.

The SINGSAMP SA z/OS sample library contains the PL/I source code for several automation routines that issue responses to selected messages. You can select the response that is most appropriate for your enterprise. You can also use them as models to create your own automation routines. The list in Table 8 summarizes these routines, the messages they respond to, and the responses they issue initially.

Table 8. SINGSAMP SA z/OS Sample Library Routines

SINGSAMP Member	Routine	Description

INGEI120	ISQI120	Responds to the following messages:
		`IEA120A Device ddd volid read, reply cont or wait.`
		`IOS120A Device ddd shared (PR volid not read.) the recovery task, reply cont or wait.`
		Issues the following response to the target: `CONT`

INGEI357	ISQI357	Responds to the following message: `IEE357A Reply with SMF values or U.` Issues the following response to the target: `U`
INGEI426	ISQI426	Responds to the following message: `$HASP426 Specify options - subsystem_id.` Issues the following response to the target: `WARM,NOREQ.`
INGEI502	ISQI502	Responds to the following message: `ICH502A Specify name for primary/backup RACF data set sequence nnn or none.` Issues the following response to the target: `NONE`
INGEI877	ISQI877	Responds to the following message: `IEA877A Specify full DASD SYS1.DUMP data sets to be emptied, tape units to be used as SYS1.DUMP data sets or GO.` Issues the following response to the target: `GO`

Table 8. SINGSAMP SA z/OS Sample Library Routines (continued)

SINGSAMP Member	Routine	Description
INGEI956	ISQI956	Responds to the following message: `IEE956A Reply - ftime = hh.mm.ss, name = operator,reason = (ipl,reason) or u.` Issues the following response to the target: `U`

The SA z/OS automation table entries in the ISQMSG0 member of the SINGNPRM data set include inactive entries that call these automation routines. To incorporate these routines into your automation, do the following:

1. Remove the comments from the corresponding automation table entries for the messages that initiate the automation routines you want to use. If you perform these steps as part of the initial SA z/OS installation, make these changes before you incorporate the SA z/OS entries. If you do this after the initial SA z/OS installation, change the NetView automation table.
2. Code the routines you will be using to issue the responses you want.
3. Compile the PL/I source code for the routines you want to use, and link the resulting object code to your PL/I library.
4. Recycle the NetView program to activate the new entries.

For automation processing to occur, each message in the NetView automation table at the focal point system and at each target system must be made available to the system's NetView program. In z/OS, MPF controls message availability to the NetView program. Examine the MPF list member in the SYS1.PARMLIB data set to ensure that the necessary messages are marked for automation. For target systems using other operating systems, check the message suppression facilities used on those systems.

Testing Messages

SA z/OS provides a collection of NetView automation table entries for your SA z/OS configuration. NetView automation table entries are in the AOFCMD member of the SA z/OS SINGNPRM installation data set. When these entries are moved to your NetView automation table, they may need additional editing.

For example, you may already test for a particular message in your production NetView automation table. If you add an entry that tests for that same message, your automation table will not run as you expect. After a match with the test criteria is found, the search of the automation table is aborted. The second NetView Automation Table statement is not found. Consequently, the message does not drive all of your required actions.

To avoid this, combine entries into a single test condition. This ensures that all required actions are scheduled for all messages. For the following message:

```
IEA320A RESPECIFY PARAMETERS OR CANCEL
```

your NetView automation table may already have the following entry: (**1**)

```
IF      MSGID = 'IEA320A'
THEN    EXEC (CMD('USERJOB') ROUTE( ONE *) ) CONINUE(Y);
```

With SA z/OS installed, the following message appears when forwarded from a PC

```
ISQ900I TSCF30 A IEA320A RESPECIFY PARAMETERS OR CANCEL
```

With SA z/OS installed, the following message appears when forwarded from zSeries or 390-CMOS processor hardware:

```
ISQ900I TAR30 OC IEA320A RESPECIFY PARAMETERS OR CANCEL
```

After the SA z/OS entries are added, the NetView automation table includes the following entry:

```
IF      TEXT = . 'IEA320A RESPECIFY PARAMETERS' .
        & MSGID = 'ISQ900I' .
THEN
        EXEC (CMD('ISQI320 '   ) ROUTE( ONE *) )
        DISPLAY(N) NETLOG(Y);
```

In this case, the first entry satisfies the IF test and the command USERJOB runs (**1**). The second command, ISQI320, is not scheduled to run because once the message matches a table entry, the autotask stops searching. Combine these two entries into a single entry, such as:

```
IF      TEXT = . 'IEA320A RESPECIFY PARAMETERS' .
        & MSGID = 'ISQ900I' .
THEN
   EXEC(CMD('ISQI320 ' ) ROUTE( ONE *) )
   EXEC(CMD('USERJOB ' ) ROUTE( ONE *) )
   DISPLAY(N) NETLOG(Y);
```

When you use the second example, both commands are scheduled.

If your NetView automation table tests the text of SA z/OS messages, the message format must match the character case for which you test. This can be done by requiring all sites to use the same format for their messages, or by duplicating AT entries in uppercase and in mixed formats.

Building the New Automation Definitions

When you are finished using the customization dialog to add message response and automation operator information to the automation policy, you need to build the system operations control files. The complete description of how to build and distribute these files is provided in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

The SA z/OS build function will place the new automation definitions in the data set defined in the Build Parameters panel.

Copy the new automation definitions into the SA z/OS NetView DSIPARM concatenation in the NetView startup procedures, or concatenate it to the NetView DSIPARM data set.

Loading the Changed Automation Environment

To reload the AMC file, automation control file and the AT perform the actions described in “Step 7: Reload MPF List and Automation Configuration Files” on page 3.

Using Pipes and ISQCCMD for Synchronous HW Commands

The System Automation for z/OS Hardware interfaces command, ISQCCMD, available for Processor Operations SNMP connections and with imitations for BCP Internal Interface connections, allows the management and control of processors and logical partitions, as well as hardware activation profiles. When used in REXX scripts, ISQCCMD provides an easy-to-use interface to automate processor operations management and configuration tasks.

The following HW commands return all their response information immediately to NetView on command completion and are therefore called *synchronous* commands:

- CCNTL
- CPCDATA
- GETCLUSTER
- GETISTAT
- GETIINFO
- GETSINFO
- GETSSTAT
- ICNTL
- PROFILE

For SNMP and BCPii connections, ISQCCMD supports NetView PIPES. On completion of the ISQCCMD command, a PIPE KEEP with the name ISQ.SNMP contains the immediate command response of the HW command that was issued, for example:

```
* ISQCCMD  G14 GETSINFO
| ISQ417I GETSINFO STATUS(SUCCESS)
| ISQ900I G14.KEY3 SC AOFA0017 GETSINFO G14 STATUS(OPERATING) PDATA(TYPE(2084)
  ,MODEL(B16),S/N(000020016F7A)) MODE(LPAR) APROF() CPCSNAME(IBM390PS.G14) NAME(G14)
  TSTIME(070825131936)
| ISQ419I ISQCCMD GETSINFO processing on G14 is complete.
* IPSFO    PIPE KEEP ISQ.SNMP | CONS
| IPSFO    AOFA0017 GETSINFO G14 STATUS(OPERATING)
           PDATA(TYPE(2084),MODEL(B16),S/N(000020016F7A)) MODE(LPAR) APROF()
           CPCSNAME(IBM390PS.G14) NAME(G14) TSTIME(070825131936)
```

In the example above, the HW common command GETSINFO was issued at a NetView console. Embedded in the 'ISQ' messages the response from the hardware is displayed on the console, starting at report ID AOFA0017.

The same information is available if you reference the PIPE KEEP with the name ISQ.SNMP, once the ISQCCMD command completed, as shown in the example, with the content of ISQ.SNMP displayed on the console.

In a REXX script, this can be coded as shown in the following example:

```
/*ReXX*/
/* Display CPC information using the ISQ.SNMP KEEP */
Arg cpcname
'ISQCCMD 'cpcname' GETSINFO'
If RC = 0 Then Do
  'PIPE KEEP ISQ.SNMP ' ,
  '  | LOC /AOFA0017/ ' ,
  '  | LOC /'cpcname'/' ,
  '  | CONS ONLY'
End
```

As an alternative, you can get the immediate ISQCCMD HW responses directly into the PIPE input stream if you use the PIPE NETVIEW stage followed by an EXPOSE TOTRAP stage. In this case, all ISQ messages and the AOFA0017 report data is available for PIPE processing.

```
/*ReXX*/
/* Display CPC information in a PIPE */
Arg cpcname
'PIPE NETV ISQCCMD 'cpcname' GETSINFO' ,
'  | EXPOSE TOTRAP '  ,
'  | LOC /ISQ90/      ,   /* takes ISQ901I or ISQ900I */
'  | LOC /AOFA0017/   ,
'  | LOC /'cpcname'/' ,
'  | CONS ONLY'
```

Automating Asynchronous Hardware Commands with ISQCCMD and PIPES

The following ISQCCMD hardware commands return two messages to NetView. First a message that the HW command has either been accepted for execution or rejected. Second, if an acceptance message was issued, a completion event message that contains the actual success or failure information of the command is sent asynchronously.

- ACTIVATE
- CBU
- CTRLCONS
- DEACTIVATE
- EXTINT
- LOAD
- OOCOD
- RESERVE
- RESTART
- START
- STOP
- SYSRESET

Automation scripts using the ISQCCMD interface must distinguish between the accepted or rejected response of an asynchronous HW command and the actual command completion information, which may either indicate successful execution or a failure. The asynchronous command completion events from the hardware are made available for message automation and TRAP AND WAIT processing by ProcOps. Application scripts using the ISQCCMD interface can get the Accepted or Rejected responses directly at ISQCCMD termination time. The Accepted response can then be used to wait for the command completion event message.

Member INGEI004 of the SINGSAMP library provides a REXX sample illustrating how asynchronous hardware commands can be automated using ISQCCMD and NetView PIPES, together with TRAP and WAIT.

VM Second Level Systems Support

This feature provides ProcOps support to control and monitor guest machines running under VM.

ProcOps allows an operating system to be IPLed into a processor, amongst other facilities. Such an operating system is VM. Within VM other operating systems can be IPLed as guest machines. Of particular interest are LINUX guest machines, but MVS, VSE and even VM guest machines may be possible. (Lower levels of guest machines are not considered). Previously there was no effective way to enter commands to and receive messages from such a guest target system in order to validate that it had IPLed correctly, or that it is behaving correctly.

With second level guest machine support you can:

- Capture messages issued by the guest machine itself and route these back to the ProcOps process for display or automated processing, or both
- Send commands to the guest machine from ProcOps, either as operator requests or automated actions

Guest Target Systems

The most likely guest machine that is used as a target system is a LINUX system. When a LINUX machine has a secondary user, the secondary user can use CP SEND commands to:

- Issue CP commands to the guest machine
- Log on as a user to LINUX
- Enter LINUX commands (after logging on)

(It is also possible to set up the LINUX system in such a way that LINUX commands can be entered on the VM console without logging on to LINUX.)

The secondary user receives:

- All ″boot up messages″
- Responses to CP commands that are run on the guest machine
- Responses to logon and LINUX commands

MVS machines are more complex. When an MVS machine is running, the original VM user first becomes an NIP console and then an MCS console. In these console modes MVS takes over all I/O to and from the console, and MVS messages to it cannot be intercepted by any CP facilities. Hence the SCIF SEND command cannot be used to send commands to MVS, nor can MVS messages to this console be intercepted.

However a ″virtual SCLP console″ for the guest machine can be used. During the NIP phase of initialization, use of this console can be forced by configuring the guest virtual machine so that it has no usable 3270 consoles. NIP then directs its messages to the guest machine as line mode commands. This is analogous to the stream of messages sent to the Operating System Messages (OSM) window on an HMC by an MVS system running in a logical partition.

Responses to any NIP messages are entered using the CP VINPUT command. Internally this is done when an ISQSEND command is issued to the operator console (OC) of the target system. To ensure that such VINPUT commands are processed correctly, the guest machine must be operating in RUN ON state at this time.

To ensure that RUN ON state is set, a CP SET RUN ON command is sent to all MVS guest machines at the time when the guest machine is started by the PSM.

Once MCS operation is established, important messages requiring operator action are directed to the guest machine. Again, these are analogous to the stream of messages directed to the OSM window of the HMC. Initially, commands cannot be entered to MVS. To do so, it is necessary to enter "Problem Determination Mode". To enter this mode, a VARY CONSOLE(*),ACTIVATE command must be entered. Once this is done:

- All MVS messages that are displayed are routed to the guest machine
- Commands may be entered using the CP VINPUT command.

Problem Determination is not generally recommended.

To enter LINUX commands it is normally necessary to log on to LINUX. This requires a user ID and a password. So, to provide for LINUX commands would require the specification of a user ID and a password to ProcOps, with all the attendant difficulties in the area of security. At present the LINUX system is considered IPL COMPLETE when specified messages have appeared. These do not require a user logon.

VM machines may also be guest machines. Third level guest machines are not supported.

VSE machines may also be guest machines.

Customizing Target Systems

LINUX

The LINUX target system should have in its VM Directory entry, a CONSOLE statement that sets its PSM as its default secondary user. For example, if the virtual machine LNXAO1 is controlled by a PSM running in virtual machine ISQPSM1, then its CONSOLE statement might be:

```
CONSOLE  009 3215 T ISQPSM1
```

When a LINUX target system is to be deactivated a FORCE command is used to shut it. The default guest signal timeout interval values (set by the SET SIGNAL command) and values defined for the guest machine determine the interval used when allowing the LINUX system to shut in an orderly fashion. If this function is required for a guest, you must ensure that this is set accordingly.

Such actions may include updating the etc/inittab entry on the LINUX system itself, and setting up a SHUTTRAP module on the VM host.

MVS

This too should have a CONSOLE statement in its VM directory entry that defines its PSM as its secondary user:

```
CONSOLE  01F 3270 T ISQPSM1
```

It should also IPL a CMS system as its initial action. Once this CMS system is IPLed it should run a PROFILE EXEC that includes the statements similar to the following:

```
SET RUN ON
DETACH   01F
IPL   7700
```

The SET RUN ON is needed so that when a response is to be sent to a NIP console the VINPUT command used is effective.

The DETACH is used so that when the MVS system IPLs it will find none of its defined 3270 consoles available to it. (You should also ensure that no user issues a VM DIAL to an address that is defined as a NIP or MCS console.)

The IPL command is used to IPL the MVS system.

The MVS system itself should have included in its active CONSOL*xx* definition a CONSOLE statement for the SYSCONS so that commands can be entered to MVS after it is IPLed, for example:

```
CONSOLE DEVNUM(SYSCONS)
        ROUTCODE(ALL)
        AUTH(MASTER)
        MSCOPE(*)
        CMDSYS(*)
        UD(Y)
```

VM

This too should have a CONSOLE statement in its VM directory entry that defines its PSM as its secondary user:

```
CONSOLE  01F 3270 T ISQPSM1
```

It should also IPL a CMS system as its initial action. Once this CMS system is IPLed it should run a PROFILE EXEC that includes the statements similar to the following:

```
SET RUN ON
DETACH   01F
IPL   7700
```

The SET RUN ON is needed so that when a response is to be sent to a console the VINPUT command used is effective.

The DETACH is used so that when the VM system IPLs it will find none of its defined 3270 consoles available to it. (You should also ensure that no user issues a VM DIAL to an address that is defined as a Operator Console)

The IPL command is used to IPL the VM system.

The VM system itself should include within its OPERATOR_CONSOLES statement in the SYSTEM CONFIG file (which resides on the ″parm disk″) a specification for the emulated system console, for example:

```
OPERATOR _CONSOLES    01F    020    System_Console
```

This ensures that when VM IPLs and finds no regular consoles available, it then uses the emulated system console. This in turn directs the messages to the secondary user as a stream of line-mode messages.

VSE

This too should have a CONSOLE statement in its VM directory entry that defines its PSM as its secondary user:

```
CONSOLE  01F 3270 T ISQPSM1
```

It should also IPL a CMS system as its initial action. Once this CMS system is IPLed it should run a PROFILE EXEC that includes the statements similar to the following:

```
TERM CONMODE 3215
IPL   7700
```

The `TERM CONMODE 3215` command sets the console into line mode.

Chapter 9. How to Automate USS Resources

Note: USS tasks behave differently when started as STCs rather than directly in the USS environment.

When started as an STC, the starting user ID may differ so that the AOFUXMON monitor routine is in most cases not able to internally trigger ACTIVMSG UP=YES.

Thus it is much simpler for automation to start these applications with INGUSS. There is then no AT entry required for the UP message. SA z/OS is able to internally simulate this so that you do not have to worry about UP messages.

Job names (at least the last character of the jobname) are not predictable for USS resources. However, AOFUXMON is able to handle this by monitoring the path within USS and changing the defined job name in SA z/OS accordingly. For the syslog daemon you would define the job name as SYSLOGD. When the application is started and changes the jobname to, say, SYSLOGD7, AOFUXMON adjusts the SA z/OS data model to reflect this.

This cannot be handled in the AT with a generic entry for SYSLOGD* because the change in job name is caused by the USS process that creates a new address space with the new name, whereby the 'old' address space with the 'old' name terminates. This means that you get an ended message for the old address space and an up message for the new address space. Again the sequence of these messages is unpredictable.

Integration of z/OS UNIX System Services

The following functions are supported by SA z/OS for z/OS UNIX applications:

- Starting and stopping of applications
- Monitoring of:
  - Processes (represented by the command or path and user ID)
  - TCP Ports
  - Files and file systems
  - Generic User Monitoring (the user supplies a z/OS UNIX monitoring routine or script)
- Using an API to execute z/OS UNIX commands (INGUSS command)

Infrastructure Overview

The z/OS UNIX resources that should be automated must run in the z/OS UNIX of a z/OS system that is already automated by SA z/OS. From the automation manager's perspective the NetView agent of this system is responsible for the z/OS UNIX resources.

For command execution through INGUSS or user-defined monitoring, a z/OS UNIX program (provided by SA z/OS) is directly invoked by SA z/OS. This program (*ingccmd*) executes UNIX commands and runs when started by SA z/OS with the jobname INGCUNIX. ingccmd is the extension of the NetView-based

agent into z/OS UNIX. To monitor the standard z/OS UNIX resources (process, port, or file) an SA z/OS internal routine is started.

Setting Up z/OS UNIX Automation

Customization of z/OS UNIX Resources

z/OS UNIX resources are introduced to SA z/OS by defining them in the SA z/OS customization dialogs.

The customization dialogs support the application type USS. If USS is selected, you can enter z/OS UNIX-specific data such as a UNIX user ID, command or path, filename, or monitored port. Choose one of these fields to enter the data.

The start and stop definitions can be varied between MVS and z/OS UNIX commands. For example, to stop an application you can issue a UNIX kill command first and (if this was not successful) you can perform an MVS cancel later.

Definitions for Automation Setup

The HFS path where the program shipped with SA z/OS is located must be defined in the SA z/OS setup panel. When user-defined UNIX monitoring is used and no absolute path is specified for the monitoring routine, SA z/OS tries to start the user-defined monitoring routine in this directory.

Definitions for z/OS UNIX Resources

To define a new application entry (APL, class, or instance), specify the application type USS on the Define New Entry panel. When choosing the application type USS, the option USS Control is displayed on the Policy Selection panel.

When selecting USS Control on the Policy Selection panel, you can enter the data for the new z/OS UNIX resource. For a class only the user ID and the z/OS UNIX monitoring routine can be specified on this panel. All other definitions (for example, from/to, dependencies, etc.) can be entered as usual.

USS applications must be defined with a HASPARENT relationship to JES.

For object type INSTANCE you can define whether this resource is one of a process, a TCP port, or a file or file system, as shown in Figure 11.

```
Monitoring Command. . .
 _____________________________________________________________________
 _____________________________________________________________________

 Enter or update one of the following fields:

 Command/Path. . .
 /u/user/usstest/usstest
 _____________________________________________________________________
 _____________________________________________________________________

 File Name . . . .
 _____________________________________________________________________
 _____________________________________________________________________

 Monitored Port. . _____
```

Figure 11. z/OS UNIX Control Specification Panel for Type INSTANCE

Note: There are two monitoring routines:

- AOFUXMON, which is called by SA z/OS for UNIX System Services resources. (This must always be specified.)
- A program in the HFS that is entered in the *Monitoring Command* field of the z/OS UNIX Control Specification panel, and is called by AOFUXMON. This means that if you specify this monitoring command, you also have to specify AOFUXMON.

If this program does not begin with a "/" it must reside in the same directory as the SA z/OS-supplied z/OS UNIX routine ingccmd. Otherwise the name specified is considered to be an absolute path identifier.

The UNIX monitoring routine must have an exit value. It can be one of the following:

0 Resource is available
4 Resource is starting
8 Resource is unavailable
12 Error occurred

If the user-specified monitoring routine loops, it will receive a SIGKILL after the AOFUSSWAIT time (defined in stylesheet CNMSTGEN).

Hint:
It is possible to write a message from this UNIX monitoring routine to the MVS system log, in order to trigger an action or perform a status change through the NetView Automation Table (AT).

The monitoring routine AOFUXMON must be specified, otherwise the default monitoring routine (usually INGPJMON) will be called, which is not sufficient for z/OS UNIX resources.

The *Job Type* field can be either MVS or NONMVS:

MVS Is only used for resources that represent a process with a unique jobname. For these resources SA z/OS accepts the following messages for status changes:

- IEF403I Job started
- IEF404I Job ended
- IEF450I Job abended

If no start command is specified, the default MVS start method (`s <JOBNAME>`) is used.

NONMVS
SA z/OS ignores the messages listed above for status changes. This is necessary if the jobname is not unique.

For z/OS UNIX resources the Start Timeout interval begins when SA z/OS issues a start command for an application. After the start timeout the monitoring method is triggered. When the monitor detects the resource as available, the agent status is set to 'ACTIVE'. After another start timeout interval and successful monitoring, the ACTIVMSG generic routine is triggered which sets the agent status to 'UP'. The default value for Start Timeout is 2 minutes.

Automated Resources

Process Monitoring: No UNIX process identifiers (PIDs) can be monitored. The monitoring routine needs the start command and the user ID that the process belongs to. This information can be obtained with the UNIX command `ps`. In the following example all processes belonging to user USER are displayed:

```
USER:/u/user/ingcmd>ps -e -o comm
COMMAND
/bin/sh
/usr/sbin/rlogind2
/bin/ps
/bin/sh
/usr/sbin/rlogind2
USER:/u/user/ingcmd>
```

This means that automation could not distinguish between the two processes started by `/usr/sbin/rlogind2`. Processes started by identical commands must have different user IDs.

If it is necessary to automate processes running multiple instances, a user could use softlinks to distinguish between the different processes. For example, the process:

```
/u/user/usstest/testme
```

should be started more than once. In this case, create some softlinks:

```
USER:/u/user/usstest> ln -s testme test1
USER:/u/user/usstest> ln -s testme test2
```

This results in:

```
USER:/u/user/tt>ls -al
total 216
drwxrwxr-x   2 USER     DE#03243    8192 Jan 24 16:24 .
drwxr-xr-x  19 USER     DE#03243    8192 Jan 24 16:23 ..
lrwxrwxrwx   1 USER     DE#03243       6 Jan 24 16:24 test1 -> testme
lrwxrwxrwx   1 USER     DE#03243       6 Jan 24 16:24 test2 -> testme
-rwxrwxr-x   1 USER     DE#03243   94208 Jan 24 16:23 testme
```

These three programs (being the same ″real″ program) can be automated with the three different start commands test1, test2, and testme. These links may be created as a prestart command and deleted as a shutfinal command.

Note: Only the command is used, not the parameters that were used to start the program. This is because a program may be started by SA z/OS with different startup parameters, depending on what the automation manager told the automation agent to do. In this case, the only constant value is the command, not the parameters.

TCP Port Monitoring: Exactly one TCP port number can be entered for one resource. SA z/OS monitors the local host as returned by the function `gethostid()`. When this port has a state of 'listening,' this resource is considered to be 'available' in terms of SA z/OS. All other states of the port will map to 'unavailable.'

File or File-System Monitoring: The existence of a file (belonging to a certain user) is verified. Many applications create files at startup and delete these files when terminating normally. If more than one file should be monitored, this can be modeled as an application group (APG) in the automation manager.

This monitoring can be used to determine if a certain file system is mounted. The start command for this resource would be a UNIX 'mount' command, the stop command a UNIX 'umount'.

Start and Stop Definitions (INGUSS Command)

If the resource is to be controlled by traditional MVS commands, this could be done in the same way as for all other MVS applications. Issuing commands in the z/OS UNIX environment is done by specifying the INGUSS command at the start or stop definitions.

To issue commands in the USS environment use the INGUSS command (for more details see *IBM Tivoli System Automation for z/OS Programmer's Reference*).

Note: INGUSS can only be used if the primary JES is available. Therefore, z/OS UNIX resources using INGUSS need a HASPARENT dependency to JES. Most z/OS UNIX applications have this dependency. If you want to issue prestart commands, an additional PREPAVAILABLE dependency is necessary.

z/OS UNIX and MVS commands can be mixed in different shutdown passes.

Command Examples:

Start Command for a Process: To start a process with the command and jobname specified in the customization dialogs, enter `INGUSS JOBNAME=&SUBSJOB &SUBSPATH` on the Startup Command Processing panel, as shown in Figure 12.

```
Type         Automated Function/'*'
Command text

INGUSS JOBNAME=&SUBSJOB &SUBSPATH
```

Figure 12. Startup Definition for a Process

Only the command that was used to start an application or a process can be monitored. If the same program is to be started multiple times, a softlink as prestart command could be used to distinguish the processes.

Use a Softlink to Distinguish Processes That Run the Same Executable File as Prestart Command: Figure 13 shows an example to create a softlink for &SUBSPATH (the path parameter of the resource issuing the command, for example, `/u/user1/uss1`) and link to the file `/u/user1/usstest`.

```
Type         Automated Function/'*'
 Command text
             *
INGUSS /bin/ln -s /u/user1/usstest &SUBSPATH
```

Figure 13. Creating a Softlink

When looking at the HFS, this results in:

```
USER1:/u/user1>ls -l
total 408
lrwxrwxrwx   1 USER1     DE#03243       7 Feb 13 12:44 uss1 -> usstest
-rwxrwxr-x   1 USER1     DE#03243  163840 Jan 29 14:55 usstest
```

Stop Commands for a Process: An z/OS UNIX process may be stopped in different ways (escalation passes). For example, you can first use the z/OS UNIX `kill` command, if that does not work use z/OS UNIX `kill -9`, and finally enter an `MVS cancel` command.

Enter the definitions for this example as shown in Figure 14.

```
Pass         Automated Function/'*'
Command Text
1
INGUSS /bin/kill &SUBSPID
_____________________________________________________________________
_____________________________________________________________________

3
INGUSS /bin/kill -9 &SUBSPID
_____________________________________________________________________
_____________________________________________________________________
4
MVS C &SUBSUSSJOB,A=&SUBSASID
_____________________________________________________________________
_____________________________________________________________________
```

Figure 14. Stop Definitions for a Process

&SUBSPID is replaced at run time by the real PID of the process.

Stop Command for a File: A stop command for a file may be deleting the file. The filename entered in the customization dialogs can be found in &SUBSFILE, as shown in Figure 15.

```
Pass         Automated Function/'*'
 Command Text
 1
 INGUSS /bin/rm &SUBSFILE
 _____________________________________________________________________
```

Figure 15. Delete a File

Example: inetd

The inetd is the UNIX internet daemon. It allows you to invoke several others and it should be started at IPL time (normally through `/etc/rc`). It then listens for connections on certain internet sockets. Its configuration file is `/etc/inetd.conf`

The following is a sample inetd configuration file:

```
login    stream  tcp  nowait  OMVSKERN  /usr/sbin/rlogind rlogind -m
exec     stream  tcp  nowait  OMVSKERN  /usr/sbin/orexecd orexecd -d
otelnet  stream  tcp  nowait  OMVSKERN  /usr/sbin/otelnetd otelnetd -k -t
daytime  stream  tcp  nowait  OMVSKERN  internal
time     stream  tcp  nowait  OMVSKERN  internal
netbios-ssn stream tcp nowait OMVSKERN  /local/samba/bin/smbd smbd
```

When a service request is detected at one of its sockets, it decides what service the socket corresponds to and invokes a program to service the request. Then it normally continues to listen on the socket the last request came in at (see Figure 16 on page 105).

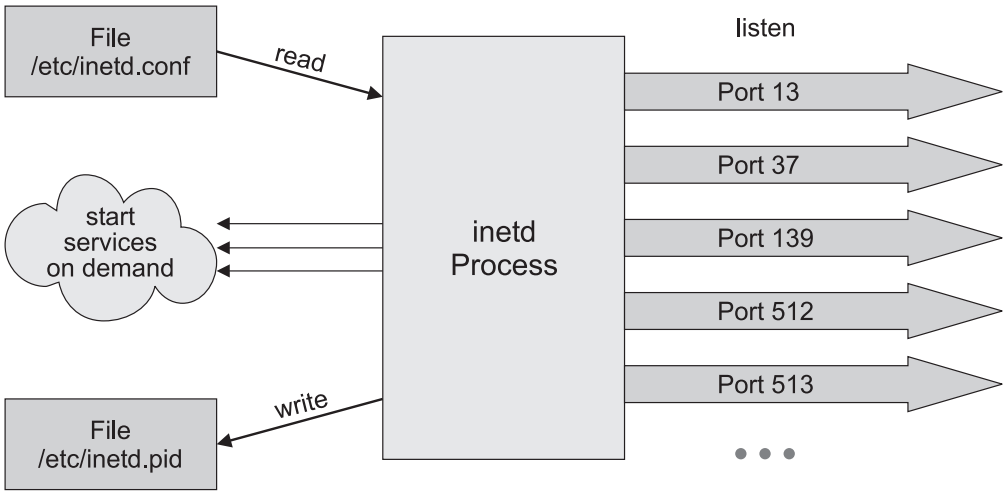


Figure 16. Structure of inetd

The inetd started with the configuration file above will listen on the following sockets:

```
USER:/etc>netstat -a | grep INET
INETD1   00006B80 0.0.0.0..13           0.0.0.0..0            Listen
INETD1   00006B7D 0.0.0.0..513          0.0.0.0..0            Listen
INETD1   00006B7E 0.0.0.0..512          0.0.0.0..0            Listen
INETD1   00006B7F 0.0.0.0..623          0.0.0.0..0            Listen
INETD1   00006B82 0.0.0.0..139          0.0.0.0..0            Listen
INETD1   00006B81 0.0.0.0..37           0.0.0.0..0            Listen
```

Whereas the services and the real port numbers correspond according to `/etc/services`:

```
daytime           13/tcp           #Daytime
time              37/tcp timserver #Time
netbios-ssn      139/tcp           #NETBIOS Session Service
exec             512/tcp           #remote process execution;
login            513/tcp           #remote login a la telnet;
otelnet          623/tcp           #OE telnet
```

The UNIX internet daemon (inetd) can be defined in the customization dialogs, for example:

```
Application Name: INETD/APL   Application Type: USS
Command/Path: /usr/sbin/inetd   User ID: OMVSKERN
Port: -      File:



Application Name: INETFILE/APL  Application Type: USS
Command/Path:      User ID: OMVSKERN
Port: -      File: /tmp/inetd.pid



Application Name: INETPORT/APL  Application Type: USS
Command/Path:      User ID: OMVSKERN
Port: 513      File:
```

Define a basic group containing all resources with relationships which indicate that:

- The file is created by the inetd process and can never be started or created directly by SA z/OS.

- The inetd process listening on the port can never be started or created directly by SA z/OS.

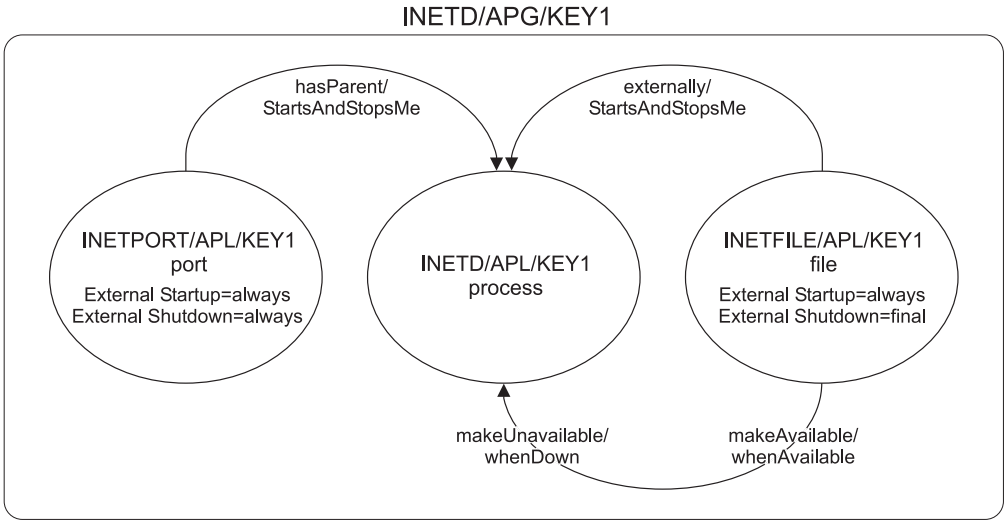


Figure 17. Dependency Graphic for inetd

The example above recognizes the inetd (modeled as a group) as up and running when the process `/usr/sbin/inetd` started by user OMVSKERN shows up, the file `/tmp/inetd.pid` exists and port 513 is in status 'listen' (inetd will listen to this port for incoming login requests).

You can only choose a port that is defined in `inetd/conf`.

Start definition for INETFILE/APL
None.

Start definition for INETPORT/APL
None.

Start definition for INETD/APL
CMD: `INGUSS JOBNAME=&SUBSJOB &SUBSPATH /etc/inetd.conf`

(&SUBSPATH is substituted at run time by the parameter command/path.)

Stop definitions for INETFILE/APL
CMD: `INGUSS /bin/rm &SUBSFILE`

(This will remove the file if not yet removed by the inetd process.)

Stop definition for INETPORT/APL
None.

Stop definitions for INETD/APL
```
CMD: INGUSS /bin/kill &SUBSPID
CMD: INGUSS /bin/kill -9 &SUBSPID
CMD: MVS C &SUBSUSSJOB,A=&SUBSASID
```

&SUBSPID will be replaced by the z/OS UNIX command routine with the real PID that matches the parameter's command/path and user ID. In the following example this is 33554821:

```
USER:/u/user/ingcmd>ps -e -o pid,comm -u OMVSKERN
       PID COMMAND
  33554481 /bin/sh
  50331698 /usr/sbin/rlogind2
```

```
33554486 /usr/lpp/netview/bin/cnmeunix
67108927 /bin/sh
83886176 /bin/ps
33554821 /usr/sbin/inetd
83886472 FTPD
67109276 /bin/sh
16777629 /usr/sbin/rlogind2
33554924 HSAPYTCP
```

Hints and Tips

Trapping UNIX syslogd Messages

To trap UNIX syslogd messages, an entry must be added to the syslogd configuration file `/etc/syslog.conf` in order to forward the messages to the MVS system log. Thus, messages can be processed by the Automation Table (AT).

To forward all messages to the MVS log add the following entry:

```
*.* /dev/console
```

To send special messages to the MVS log only, follow the syslog message naming guidelines (for example, for warning messages use `*.warn`). `/dev/console` can be used as an ordinary file to write to.

The UNIX messages have the MVS message ID BPXF024I and are multiline messages.

Figure 18 shows an example of a UNIX message:

```
M 13:45:21.34 STC03602 00000090 BPXF024I (USER) Feb 13 13:45:21 BOEKEY1 syslogtest 67109100 : This is
S                              498
D                  498 00000090 a test message
```

Figure 18. Example of a UNIX Message

Debugging

Debugging can be activated for z/OS UNIX monitoring and command execution on the AOCTRACE panel. The clist for monitoring is AOFUXMON and for command execution AOFRSUSS.

Turning on debugging for AOFRSUSS implicitly turns on debugging for ingccmd (the SA z/OS command server).

The debugging messages will be written to the netlog and to the z/OS UNIX system log (syslogd).

Chapter 10. How to Enable Sysplex Automation

This chapter describes enhancements to Parallel Sysplex® automation, how to use the SA z/OS customization dialogs to enable them, and how to customize your system.

Note: If you use a host code page other than 037, the hexadecimal representation of the at sign (@) can be different. Use the letter represented by the hex code X'7C' for the at sign.

Sysplex Functions

The following functions are described:

Managing Couple Data Sets

Couple data sets (CDSs) contain control information about the sysplex and its resources, and are of crucial importance for the functioning of a Parallel Sysplex. Particularly important are the SYSPLEX couple data set, which contains information about the systems and the communication structure (XCF groups) of the sysplex, and the CFRM couple data set, which specifies its coupling facilities (CFs) and structures (see "Managing Coupling Facilities" on page 111). Every MVS system in a Parallel Sysplex must have access to these CDSs, and to those of all other implemented sysplex functions, such as SFM and Application Response Measurement (ARM).

If a member system cannot access a CDS, the corresponding sysplex function is impacted, and in some cases the sysplex will go down. It is therefore recommended that you define two CDSs to XCF for every CDS type required for the implementation of the sysplex. One of these, the *primary* CDS, is the one that is actually used. The other, which is called the *alternate* CDS, serves as a backup copy. The two CDSs contain the same data. Whenever the primary CDS changes, XCF updates the alternate CDS accordingly. If an alternate CDS is available for a certain type, XCF automatically switches to this alternate CDS whenever a member can no longer access the primary CDS.

All CDSs except the sysplex couple data set contain one or more user-defined configurations, called *policies*. For each CDS type, only one policy can be active. However, it is possible to switch the active policy at run time. Refer to *IBM Tivoli System Automation for z/OS Operator's Commands* for further information about the INGPLEX command.

SA z/OS offers two functions for easier CDS management:

- Automated creation and recovery of alternate couple data sets for continuous availability
- INGPLEX CDS, which simplifies management of couple data sets

Ensuring Continuous Availability of Couple Data Sets

When an alternate CDS exists for a given CDS type and the current primary CDS fails, XCF makes this alternate the primary CDS. After this switch, however, an alternate CDS no longer exists, and if the current primary CDS also fails, the problems that were to be avoided by the creation of an alternate occur again. To avoid this single-point-of-failure situation, SA z/OS provides a recovery mechanism that tries to ensure that an alternate CDS is always available for every CDS type used.

SA z/OS creates a new alternate CDS in the following two situations:

- During initialization, SA z/OS checks that an alternate CDS is specified for every primary CDS. If there is a primary CDS for which no alternate CDS exists, SA z/OS automatically creates it.
- At run time, SA z/OS ensures that a new alternate is created whenever the current alternate has been removed or switched to the primary one.

Customization

Recovery of alternate CDSs is initiated either by the CDS function of INGPLEX or in the background (for example, at initialization time). Background recovery can be switched on and off by using the SA z/OS customization dialogs. Automatic re-creation with INGPLEX CDS is always enabled.

You must specify the spare volumes that SA z/OS may use for creating missing alternate CDSs (using the policy item SYSPLEX from the Policy Selection panel for sysplex groups). This is also required for automatic creation with INGPLEX CDS. Every CDS type has its own pool of spare volumes. Note that if you do not define spare volumes for a CDS type, no recovery will be performed for this type. For details on the use of the customization dialogs, see "Enabling Continuous Availability of Couple Data Sets" on page 121.

You can control access to those functions of INGPLEX CDS that modify the sysplex configuration. Refer to Appendix A of *IBM Tivoli System Automation for z/OS Planning and Installation* for details.

Managing the System Logger

Terms and Concepts

The *system logger* provides a sysplex-wide logging facility. Applications that use the system logger write their log data into *log streams*. Within a Parallel Sysplex, these log streams are usually associated with a coupling facility structure. For further information about coupling facility structures, refer to "Managing Coupling Facilities" on page 111. By using a coupling facility log stream, members of a multisystem application can merge their logs even when residing on different systems.

When an application writes data to a log stream this data is stored at first temporarily in the associated structure (coupling facility log stream) or a local buffer (DASD-only log stream). From there, it is off-loaded into a log stream data set which is automatically allocated by the system logger. When this log stream data set is full, the system logger allocates a second one, and so on.

The control information for the system logger, which includes a directory for the log stream data sets of every log stream, is contained in the LOGR couple data set. The total number of log stream data sets that can be allocated by the system logger is determined when the LOGR couple data set is formatted.

Two problems that can arise in connection with the log stream data sets are a shortage of directory space in the LOGR CDS and incorrect share options for the log stream data sets. SA z/OS provides the following recovery actions for these problems:

- The primary and alternate LOGR CDSs are automatically re-sized if there is a directory shortage
- The operator is notified if the share options for log stream data sets are not defined correctly

Resizing the LOGR Couple Data Sets in Case of Directory Shortage

The LOGR CDS contains information about the log stream data sets used by the system logger. This information is stored in *directory extents*. Every directory extent record can hold information about up to 168 log stream data sets. The number of directory extents available in a LOGR CDS is specified when the CDS is formatted (`DSEXTENT` parameter). When all available directory extents are used up the system logger can no longer allocate new log stream data sets. This can cause considerable problems for applications that use the system logger.

With SA z/OS, you can avoid this situation. If you switch on logger recovery, SA z/OS automatically reformats your primary and alternate LOGR CDS with an increased DSEXTENT parameter whenever the system reports a directory shortage.

Customization

Automation of system logger recovery is enabled through the SA z/OS customization dialogs. For more details, see "System Log Failure Recovery" on page 172.

Managing Coupling Facilities

A *coupling facility* (CF) is a logical partition that provides storage for data exchange between components of an application that is distributed across different systems in a Parallel Sysplex. A Parallel Sysplex can contain more than one CF. The storage of a coupling facility is divided into areas that are called *structures*. You can imagine a structure as a special kind of data set. It is these structures, which are identified by their name, that are accessed for reading and writing by the application components.

The association between CFs and structures is dynamic. A structure that is used by an application need not be allocated at all (for example, when the application is not running), and can be allocated on different CFs at different points in time. For every structure, there exists a *preference list* that defines the CFs on which it may be allocated. The order of the CFs in that list determines which CF is selected when more than one member of the list satisfies all allocation requirements (for example, provides enough space).

The preference list, the space requirements, and other properties of the structures are defined in the active CFRM policy. This policy is contained in the CFRM couple data set. Refer to "Managing Couple Data Sets" on page 109 for further information.

XES allocates a structure that does not yet reside on any CF when an application component needs to be connected to it. Note that the application component only specifies the name of the structure that it wants to access. It is XES that decides on which CF the structure is allocated. This decision is influenced by the structure definition in the active CFRM policy. After the structure has been allocated, the

requesting application component can access it, and further components of this application can require to connect to it. An application component that has access to an allocated structure is referred to as an *active connector* to this structure.

In the simplest case, XES deallocates a structure when all connected application components have disconnected from the structure. However, an application component can require that the structure or its own connection to the structure be *persistent*. When the *structure* is persistent it remains allocated even when the application component is no longer connected to it. When a *connection* is persistent the structure remains allocated after a failure of that connection. The application component in question remains a connector to the structure, although not an active one. It is now a *failed persistent* connector. In both cases, you can force the deallocation of the structure as soon as it no longer has active connectors.

Allocated structures can be *rebuilt*. Rebuilding is the process of reconstructing a structure on the same or another CF. A rebuild consists of three main steps. First, XES allocates the new structure instance. Then, the data of the old structure is reconstructed in the new structure. Finally, XES deallocates the old structure instance. Note that you cannot specify the target CF in your rebuild request. As with structure allocation, XES selects it from the preference list.

There are two methods for rebuild: user-managed and (from OS/390 2.8 onward) system-managed. With user-managed rebuild, the active connectors are responsible for reconstructing the data. With system-managed rebuild, XES transfers the data to the new structure instance. System-managed rebuild is thus also available for structures without active connectors. These structures can either themselves be persistent or have failed persistent connections.

When an application component connects to a structure, it specifies whether it allows the structure to be rebuilt through user-managed or system-managed rebuild. For structures with active connectors, both rebuild methods require that all active connectors allow the respective rebuild method.

You can also *duplex* structures. Duplexing means maintaining two instances of the same structure on different CFs at the same time. Duplexing serves to increase availability and usability of a structure.

Typical management tasks for CFs are removing a CF from the sysplex and reintegrating it again. These tasks have several steps that must be performed in a certain order and can be quite complex. To simplify these operations, SA z/OS offers the INGCF command. INGCF has several functions, which serve to manipulate structures and the CFs themselves. For more information, see *IBM Tivoli System Automation for z/OS Operator's Commands* and the online help.

Some functions deal with the sender paths of a coupling facility. They have the following limitations. First, at least one system in the sysplex that is running the automation must know the control unit ID (CUID) of the coupling facility. If this is not the case, no missing sender paths can be resolved.

A missing sender path occurs when a coupling facility is deactivated prior to a system IPL (or reIPL) and then activated afterwards. The system that has been IPLed (or reIPLed) does not recognize the coupling facility. To determine the missing sender paths, the automation calls the HOM interface of HCD. Resolving the missing path information is only possible when either the complete network address is defined in HCD along with the processor ID, or you provide the CPC

synonym used by the automation as the processor ID. However, it is recommended that you define both. If neither is defined, the system that misses the sender paths must run the automation.

Recovery Actions

Resolving WTO(R) Buffer Shortages

When all WTO(R) buffers are in use, it is possible that commands can no longer be processed. To resolve this, there are several options: you can extend the buffer, change the properties of the affected consoles, or cancel jobs that issue WTO(R)s.

SA z/OS provides recovery of buffer shortage in two stages. It first tries to extend the buffer and modify the console characteristics, if applicable. If this does not help, it then cancels jobs that issue WTO(R)s. You must specify which jobs can be canceled by SA z/OS if there is a buffer shortage.

Customization: Automation of buffer shortage recovery is enabled using the SA z/OS customization dialogs. For more information, see "Enabling WTO(R) Buffer Shortage Recovery" on page 121.

Handling Long-Running Enqueues (ENQs)

This type of recovery is divided into the following individual functions:

- Long-running enqueue recovery
- SYSIEFSD resource recovery
- "Hung" command recovery
- Command flooding recovery

All these recoveries can be enabled and disabled individually or globally.

The long-running enqueue recovery function lets you:

- Check which resources are blocked
- Customize automation to cancel or keep the jobs that block the resource
- Customize automation to dump the jobs before they are canceled

You can determine which resources you want to monitor. You can define a value for the maximum time a job can lock a resource while other jobs are waiting for it. If this amount of time is exceeded, recovery takes place. Identification of and elimination of these potential bottlenecks helps to reduce the risk of a Parallel Sysplex outage.

While the time definition describes an inclusion list, you also have the possibility to define an *exclusion list* of resources that are not monitored at all.

For more information about enabling the ENQ function, see "Enabling Long Running Enqueues (ENQs)" on page 125.

This function has been extended by three supplementary functions:

- "SYSIEFSD Resource Recovery"
- ""Hung" Command Recovery" on page 114
- "Command Flooding Recovery" on page 115

SYSIEFSD Resource Recovery: The purpose of this function is to detect critical ENQ resources that, if held for extended periods of time, can cause commands to

hang. Hung commands often result in multisystem outages. The focus of this function is on the SYSIEFSD family of resources that are involved in 98% of hung command outages:

- SYSIEFSD Q10 – this resource is required for every command. It is used to serialize changes to the CSCB chain. If any task gets this resource and then hangs, *all commands* will be locked out of the system. This also means that *all consoles* will be locked out of the system. This is because, as soon as a console issues a command after Q10 has hung, it will be waiting behind Q10, and that locks out the task that handles all MCS consoles. EMCS consoles will then also get locked out one by one as they issue a command and also get hung behind Q10. Actions taken to free up this hang cannot include issuing a command (for example, D GRS)—the task has to be terminated via CALLRTM.
- SYSIEFSD Q4 – this resource is used to serialize changes to the UCB by allocation and VARY command processing. Allocation obtains the resource as SHARED, while the VARY command obtains it exclusively. If a VARY command hangs while holding this resource, all allocations will also hang. The VARY command that is hung can be displayed and abended with the CMDS command.

It is highly recommended that you add SYSIEFSD Q4 and SYSIEFSD Q10 to the RESOURCE DEFINITION policy item with a wait time of 30 seconds so that long running ENQ recovery will monitor and automate recovery of these resources.

"**Hung" Command Recovery:** The purpose of this function is to detect hung commands that often result in multisystem outages. We distinguish three situations:

1. Commands that inhibit other commands from completing execution
2. Commands that inhibit jobs from completing execution
3. Jobs that inhibit commands from completing execution

Automation examines ENQ contention associated with command processing and builds a list of blockers and waiters. The SA z/OS policy is then examined to see how long waiting commands and waiting jobs are allowed to wait before automated action is taken. The policy is also examined to determine what action (DUMP, NODUMP, KEEP or exclude) is to be taken against the blocking command or job, as follows:

1. When a command inhibits other commands from completing and no policy definitions exist for any of the waiting commands then no automated action will be taken.
2. When a command inhibits jobs from completing and no policy definitions exist for the blocking command then no automated action will be taken.
3. When a job inhibits commands from completing and no policy definitions exist for any of the waiting commands then no automated action will be taken.

If long-running ENQ and hung command recovery detect the same resource requires automated action at the same time, then the hung command recovery policy definitions will take precedence and hung command recovery will automate the resource.

The action taken (DUMP, NODUMP, KEEP or exclude) is identical to the long-running ENQ recovery action.

In either case only commands that are waiting on blocked resources are considered. Unlike SYSIEFSD resource recovery, which is unaffected by long-running ENQ recovery, "hung" command recovery only considers those resources that are not being monitored by long-running ENQ recovery. If

long-running ENQ recovery is disabled then all resources, even those defined as long-running ENQ resources, are considered for "hung" command recovery. It is also important to realize that if long-running ENQ recovery is enabled and a generic ″catchall″ resource definition applies, then "hung" command recovery cannot occur, because long-running ENQ recovery always take precedence.

Commands are executed by the master and console address spaces. Thus when a resource blocker is from either of these address spaces it is considered to be a blocking command rather than a blocking job.

As with resources, you can make similar definitions for commands that determine how long a command is permitted to lock a resource while other commands are waiting for the resource.

If the resource blocker is a job then recovery actions are only taken when the job has blocked the command for 3 consecutive iterations of ″hung″ command recovery processing. This results in a job blocking a command for no more than 90 to <120 seconds.

Recovery action for the blocking job or the job that issued the blocking command is the same as that specified for long-running ENQ recovery automation.

Command Flooding Recovery: The purpose of this function is to detect jobs that flood a command class. Command flooding can cause log buffer shortages and inhibits other commands from executing. Both can lead to a multisystem outage.

When all (50) TCBs that are reserved for command processing are in use, new commands are queued to the waiting queue. In this case the system issues message IEE806A which triggers this function to evaluate what jobs are causing the situation.

Jobs that just issue a set of commands, such as 200 (or more) ″VARY dev,ONLINE″ commands should *not* be considered during the evaluation. This is achieved by comparing the current and the previous snapshot of the affected command class.

Snapshot processing is scheduled when message IEE806A is trapped. The interval time between the snapshots is 3 seconds by default (see “Enabling Long Running Enqueues (ENQs)” on page 125 for details about adjusting this value if necessary). The interval should give these jobs enough time to finish issuing commands before the first snapshot is taken. Only jobs that issue commands on two consecutive snapshots become subject of the recovery action.

Before the recovery action takes place, the number of commands that are issued by the job must exceed a threshold (see below) and at least one of the commands must not be involved in a lock contention that is handled by the ″hung″ commands recovery.

The recovery action depends on the job definitions (see “Enabling Long Running Enqueues (ENQs)” on page 125). If the job can be canceled, the recovery also removes its waiting commands and terminates its executing commands. The recovery action is completed either with message ING922E or with message ING924E. The latter message is repeatedly issued approximately every minute until the waiting queue becomes empty.

The threshold is calculated by subtracting the number of jobs that are issuing commands in the command class from the total number of TCBs (50) that are reserved for command processing. This prevents jobs that repeatedly issue few commands from being evaluated .

The recovery ends when the message IEE061I is issued.

Note: The dump definitions are not in effect if a dump should be taken when the job is canceled. This is because the recovery routine of the job that is being canceled can suppress the dump.

Customization: Automation of handling long-running enqueues is enabled through the SA z/OS customization dialogs. For more details, see "Enabling Long Running Enqueues (ENQs)" on page 125.

System Removal

The purpose of this function is to isolate failed systems from a Parallel Sysplex by removing them as quickly as possible. It also ensures fast mean time to recovery (MTTR) for those system images that you wish to restart immediately if an unavoidable outage occurs.

Note: This function is unavailable when running on a z/OS image which runs under z/VM, even if the function is enabled.

In particular, the function automates the messages IXC102A and IXC402D.

The automation of the first message completes the Sysplex Failure Management (SFM). Under certain circumstances SFM cannot complete the isolation of a failed system. This is because SFM's HW isolation, resetting the channel subsystem (CSS) of the failed system, is driven through the CF. When connectivity between the system image and the coupling facility is lost, SFM cannot perform the hardware isolation (ISOLATE command) and defers resetting the system image until manual operator intervention occurs. Message IXC102A tells the operator to manually reset the HW and then reply "DOWN" to the message, after which SFM safely partitions the system image out of the sysplex. The longer the delay lasts, the more the components and applications that rely on XCF messaging are impacted. The delay can eventually lead to a sysplex outage when the failed system has I/O operations pending. Automation of this message minimizes the delay.

The second message has the same impact as the first one. However, this message indicates a possible temporary inoperative status of the system due to a missing status update. For this reason the automation gives the system the chance to recover before the removal takes place by replying "INTERVAL=*sss*" to the first occurrence of message IXC402D. The interval time, *sss*, is the failure detection interval that is displayed by the command `D XCF,CPL`.

The automation does the removal of a system in two stages. The first stage clears any pending I/O operations by sending a hardware command to the Support Element. This requires information about the software running on the hardware. Because the system issuing message IXC102A or IXC402D does not necessarily have access to the hardware of the failed system, the automation needs predefined mapping between software and hardware. Depending on this mapping, it then routes the hardware command to the system that has access to the hardware of the failed system. For information about how to do the mapping refer to "Enabling

System Removal" on page 123. For further information about the hardware requirements refer to *IBM Tivoli System Automation for z/OS Planning and Installation*.

The second stage replies to the outstanding WTOR with "DOWN" triggering the removal of the system from the sysplex.

Customization: Automation of message IXC102A is enabled through the SA z/OS customization dialogs. For more details, see "Step 3: Automating Messages IXC102A and IXC402D" on page 124.

Recovering Auxiliary Storage Shortage

With the automation of local page data sets, SA z/OS prevents auxiliary storage shortage outages by dynamically allocating spare local page data sets when needed. The function checks which jobs cause the shortage condition and whether additional page data sets can be added. If this is not possible, the job that is causing the shortage will be canceled if this has been defined.

To enable local page data set automation customize the PAGTOTL parameter (defined in one of the IEASYS*xx* PARMLIB members used during IPL). Make sure to set the PAGTOTL parameter to a value greater than the number of local page data sets currently used.

Local page data sets must be defined in the master catalog and should not be SMS-managed. It is recommended to use preallocated local data sets instead of dynamically allocated ones. This makes the process faster because formatting newly allocated page data sets is time-consuming (10sec./35MB). Each predefined local page data set should be allocated with 10% space of local page space currently used by the system. If predefined page data sets can no longer be allocated, new local page data sets will be created dynamically.

Customization: Automation of the recovery of auxiliary storage shortage is enabled through the SA z/OS customization dialogs. For more details, see "Enabling System Removal" on page 123.

Hardware Validation

This function performs cross-validation of the hardware configuration mapped out in the customization dialogs against the actual hardware configuration that is running. This information is critical to accurately control logical partitions (LPARs) on any supported CPC within the HMC/SE LAN over the BCP Internal Interface.

Hardware validation uses the CPC name, Partition name and Partition number to ensure that the LPARs defined in the customization dialogs are on the correct CPC and located on the correct partition number. However, this helps only for coupling facilities because their partition identifiers must be defined in the active CFRM policy.

For MVS images, information from the HMC/SE (such as system name and sysplex name that are stored during initialization) is used to verify the corresponding customization dialog definitions. During initialization of the automation's Hardware Command Interface and just before a disruptive request is sent to a partition, new checks are made to ensure that everything matches correctly.

Note: Only active images can be verified. For inactive images we must still rely on definitions made in the customization dialogs.

An active system in this context is a system belonging to the same sysplex as the system that runs the hardware validation, that is SA z/OS checks only systems and coupling facilities within its own sysplex.

Hardware validation runs on an SA z/OS system primarily during startup, and subsequently when changes to the definition in the customization dialogs are applied through the ACF command (ACF COLD, and ACF REFRESH when any CPC or image data has changed). The validation checks the definitions of all registered systems, that is whenever an SA z/OS system performs the hardware validation, it validates all systems and coupling facilities that are active in the sysplex at this point in time. Registered systems are systems running msys for Operations or SA z/OS that have joined the same XCF group.

The validation of active systems and coupling facilities requires that the CPCs that host the active systems must all be defined in the customization dialogs.

The data for inactive systems cannot be verified. However, these definitions are checked for consistency across all registered systems. As soon as one of these inactive systems or coupling facilities joins the sysplex or is made available for use, the validation is run for the particular image only.

Retrieving actual hardware information can take up to 5 minutes per CPC depending on the model and its LPARs. During the time that the hardware validation takes place all other hardware-related automation is either delayed or cannot be performed, depending on the type of recovery. For this reason the validation carries out ″delta″ processing. That is validating only the data that has changed. This also includes the absence of data resulting in terminating CPC connections when CPC definitions are missing that have been applied by a prior validation. The actions resulting from the validation are performed on ALL registered systems. This has two advantages:

- you don’t need to recycle NetView for changes in hardware definitions.
- you only need to make the changes available to one system.

The first part of the hardware validation triggered by the ACF command or the automation startup determines what CPC connections must be terminated and initiated, namely in this sequence. The resulting actions are performed on all registered systems. When this step has been completed successfully the image validation is performed.

The image validation collects actual hardware information, and verifies the current hardware definitions against the actual data and the definitions found on all other registered systems. It informs you if:

- A real system or coupling facility could not be validated because either actual hardware information or user definitions are not available
- The image definitions could not be evaluated because the actual hardware information is not available
- The real system or coupling facility is not active and the image definitions of some of the registered systems are different
- Any definition value has been corrected that was improperly defined or not defined at all

Changes in hardware definitions can be made available to all registered systems by simply invoking the command INGAMS REFRESH on only one of the these

systems. There is one exception: the change of the authorization token value used for the communication with a particular CPC. A change of this value requires 3 steps:

1. In the first step you must remove the particular CPC definition and then invoke the ACF command as above.
2. When the command completes successfully the next step is to change the authorization token value of the CPC at the Support Element.
3. The final step is to define the CPC again with the new token value and invoke the ACF command again.

Note: This behavior of the INGAMS command applies to the hardware definitions *only*.

The second part of the validation is triggered by either the message IXC517I that is issued when a coupling facility is made available for use, or by the automation itself when notified that a system joined the sysplex. Both trigger the automation to perform only the validation of the new system or coupling facility. Multiple occurrences of messages for the same system or coupling facility are ignored while this system or coupling facility is validated. In case of a new system, the advantage here is that the real hardware is validated before the system starts NetView and the automation. If this automation then detects no difference between its current definitions and the definitions of the other registered systems—which is the normal case—only a consistency check takes place. This check does not require any real hardware information.

Prerequisites

Hardware validation has the following prerequisites:

- All coupling facilities that are used in the sysplex must reside on a CMOS-S/390 G5 processor or higher. Only these processors return the partition identifier that is required for validating coupling facilities.
- The BCP Internal Interface must have been initialized to accept requests. Or, when unavailable, at least one other registered system must have access to the hardware. Registered systems are systems running msys for Operations or SA z/OS that have joined the same XCF group.

Note: Hardware validation is not supported on MVS systems running under z/VM.

Enabling Hardware-Related Automation

To enable the sysplex automation that SA z/OS provides for recovery actions and coupling facility management, the following definitions must be made in the customization dialog.

Step 1: Defining the Processor

Use the customization dialog to define a new processor of Entry Type PRO. The name should be the real physical name of the processor defined in HCD. For more information, refer to the online help or the section "Creating a New Processor" in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Step 2: Using the Policy Item PROCESSOR INFO

Use the Processor Information panel, to define a processor using entry type PRO.

Note: The connection type protocol must be INTERNAL
For more information, refer to the online help or the section "More about Policy Item PROCESSOR INFO" in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Step 3: Defining Logical Partitions

If the processor that you have defined runs in LPAR mode, define its logical partitions using the LPAR Definitions panel. You should define all LPARs that are physically available on your processor, together with the systems that run on them.

For more information, refer to the online help or the section "More about Policy Item LPARS AND SYSTEMS" in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Step 4: Defining the System

Define a system using entry type SYS, and the Define New Entry panel.

Note: To avoid receiving hardware validation messages during SA z/OS initialization, you should define all your systems (including your coupling facilities).

For more information, refer to the online help or the section "Creating a New System" in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Step 5: Connecting the System to the Processor

Connect this system to the processor that you defined in "Step 2: Using the Policy Item PROCESSOR INFO" on page 119 and to its logical partition (if you set the processor mode as LPAR).

Connect this system to the sysplex or standard group (see "Step 6: Defining Logical Sysplexes" and "Step 7: Defining the Physical Sysplex").

Note: MVS SYSNAME and the Image/ProcOps Name *must* be the same.

Restriction:

Usually, the MVS SYSNAME may begin with a number. However, in this case, it must be the same as the Image/ProcOps Name, which *cannot* begin with a number. Therefore, this naming restriction also applies to the MVS SYSNAME.

Step 6: Defining Logical Sysplexes

Define EACH logical sysplex (systems within the same XCF group ID) using entry type GRP with group type SYSPLEX.

Use policy SYSPLEX to enter the real physical sysplex name. You can use the same name in several SYSPLEX GRPs.

Use policy SYSTEMS to connect all systems within the same XCF group ID to the SYSPLEX GRP. A system can only be connected to one SYSPLEX GRP.

Step 7: Defining the Physical Sysplex

Define your real physical sysplex using entry type GRP with group type STANDARD.

Use policy SYSTEMS to connect all systems of your physical sysplex to the STANDARD GRP.

Enabling Continuous Availability of Couple Data Sets

Couple data sets (CDSs) contain important information about how to manage certain aspects of your sysplex. For example, the SFM CDS (sysplex failure management couple data set) defines how the system manages system and signalling connectivity failures and PR/SM (Processor Resource/Systems Manager) reconfiguration actions.

The following couple data sets are particularly important for the functioning of your Parallel Sysplex:

- The SYSPLEX couple data set, which defines the systems and the XCF groups of the sysplex
- The CFRM couple data set, which defines the coupling facilities and structures of the sysplex

It is recommended that you define alternate couple data sets for all couple data sets in your sysplex. These alternate couple data sets serve as backups when the primary CDS fails.

With the customization dialog you can specify a series of spare volumes for every CDS type, for example, SYSPLEX, ARM, CFRM. The first volume in the series is used to create an alternative CDS if one of the primary alternate CDSs fails.

In the customization dialog you define the potential alternate couple data sets using the Group entry type. Select a sysplex group, then select its policy item SYSPLEX (define sysplex policy) from the panel Policy Selection.

The Sysplex Policy Definition panel is displayed if you select policy item SYSPLEX from the Policy Selection panel for sysplex groups.

For a description of this panel refer to the online help or the section "More About Policy Item SYSPLEX" in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Enabling WTO(R) Buffer Shortage Recovery

The SA z/OS customization dialog supports the automation of WTO(R) buffer shortage recovery.

When using the MVS Component entry type (MVC), you can specify jobs that will be canceled or kept in case a WTO(R) buffer shortage is threatening. The jobs that you select for cancellation will then no longer issue WTO(R)s.

Select the MESSAGES/USER DATA policy item of a selected MVS Component policy object to display the Message Processing panel.

Enter `CODE` in the ***Action*** column and `WTOBUF` in the ***Message ID*** column

After pressing Enter, the Code Processing panel is displayed. For more information about this panel, refer to the online help or to the section "More About Policy Item MESSAGES/USER DATA" in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Enabling WTO(R) Buffer Shortage Recovery

WTO Recovery is performed when different messages are received by SA z/OS. The action taken when each of these messages is received is described inTable 9.

Table 9. WTOBUF Recovery Process

Recovery	Message	Actions in sequence	Command
WTO	IEA405E	Set console attributes.	
		If the deletion mode is not roll or wrap, set the mode to roll.	`K S,DEL=R,L=x`
		If any out-of-line display area exists, delete the status display.	`K E,D,L=x`
		If the interval between message rolls is not '*' or less than or equal to 1 second, set the interval to 0.25 seconds.	`K S,RTME=1/4,L=x`
		If the console receives messages not only from the local system and the WTO message buffer size has reached its maximum, remove the buffering systems from the list and add the local system to the list.	`V CN(x),MSCOPE=(l)`
	IEA404A	Suspend the console.	
		Requeue the messages to the hardcopy log.	`K Q,L=x`

		Vary the active console (COND=A) offline. For SMCS consoles, issue the appropriate VTAM command (OA05706).	`V {CN(x),OFFLINE`
			`|NET,TERM,LU1=x,`
			`TYPE=FORCE`
			`}`

		Cancel the job or TSO user that caused the shortage, but only when defined as a candidate during the customization.	`C {jobnm,A=asid`
			`|U=userid`
			`}`

	IEA406I	Resume the console if it was suspended and if it is not a SMCS console. (OA05706)	`V CN(x),ONLINE`
		Restore console attributes.	
		Set the deletion mode to the value before the buffer shortage occurred (OA05706).	`K S,DEL=old,L=x`
		Set the interval between message rolls to the value before the buffer shortage occurred.	`K S,RTME=old,L=x`
		Set the list from which the console is to receive unsolicited messages to the list before the buffer shortage occurred.	`V CN(x),MSCOPE=(l)`

		Increase the WTO message buffer size to minimize future shortages as follows:
		`new = min(9999`
		`,max(1500`
		`,1.2 * current MLIM`
		`)`
		`)`
			`K M,MLIM=new`

		Issue message AOF929 for permanent changes (MLIM). (OA05706.)	

Table 9. WTOBUF Recovery Process (continued)

Recovery	Message	Actions in sequence	Command
WTOR	IEA230E	Increase the maximum number of reply IDs to the maximum allowable value if the maximum number of systems in the sysplex is greater than 8 or the system runs in local mode.	`K M,RMAX=9999`

		Increase the WTOR message buffer size if the current RMAX value is greater than the current RLIM value as follows:
		`new = min(9999`
		`,max(10 + 2 * maxsys_in_sysplex`
		`,1.2 * current RLIM`
		`)`
		`)`
			`K M,RLIM=new`

	IEA231A	Cancel all jobs and TSO users that have outstanding WTORs and that are defined as candidates during the customization.	`C {jobnm,A=asid`
			`|U=userid`
			`}`

	IEA232I	Issue message AOF928 for irreversible changes (RMAX).	
		Issue message AOF929 for permanent changes (RLIM).	

Enabling System Removal

The SA z/OS Parallel Sysplex enhancements help you to resolve pending I/Os for systems being removed from the sysplex.

Because the automation must know where the system is located to send the command to the appropriate Support Element, you must use the customization dialog to define its hardware configuration.

Step 1: Defining the Processor and System

The processor and system must be defined as described in "Enabling Hardware-Related Automation" on page 119.

Step 2: Defining the Application with Application Type IMAGE

Use entry type APL to define a new application with Application Type IMAGE and subsystem name that is the same as the Image Name of the system that this application represents (as defined in "Step 4: Defining the System" on page 120).

Use entry type APL and select policy item APPLICATION INFO for your system. On the panel *Application Information*you can define a new application type IMAGE. For more information, refer to the online help or the section "Policy Items for Applications" in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Because the application has been defined as type IMAGE, the job name is set by default to the subsystem name and cannot be changed.

The Subtype, Scheduling Subsystem, JCL Procedure Name, ARM Element Name, and WLM Resource Name are forced to be blank.

Some other definitions in the policy item AUTOMATION INFO are also defaulted:

- the Job Type is defaulted to NONMVS
- the Monitor Routine is defaulted to INGMTSYS if nothing is specified
- the External Startup is defaulted to ALWAYS if the Monitor Routine is INGMTSYS

- the External Shutdown is defaulted to ALWAYS if the Monitor Routine is INGMTSYS

For more information, refer to the online help or the section "More About Policy Item AUTOMATION INFO" in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Step 3: Automating Messages IXC102A and IXC402D

You can automate messages IXC102A and IXC402D to avoid sysplex outages.

Note: The following shows examples for defining commands and codes for message IXC102A.

You can specify one of the following four hardware commands for each system in the sysplex that is automated.

- SYSRESET [CLEAR]
- DEACTIVATE
- ACTIVATE [P(image_profile_name)]
- LOAD [P(load_profile_name)] [CLEAR]

where

CLEAR	indicates that the storage will be cleared
P	specifies the profile to be used. The name can consist of up to 16 alphanumeric characters. If the parameter is omitted, the last profile is used.

Note:
The following restriction applies to the hardware commands ACTIVATE and LOAD:

Both commands invoke processor functions that can cause asynchronous events such as operator messages at BCP (Basic Control Program) Internal Interface initialization time or processor hardware wait states. Currently, the BCP Internal Interface does not allow the monitoring and control of these events.

Use policy item MESSAGES/USER DATA of the SA z/OS customization dialog to define commands and codes for message IXC102A and IXC402D. Enter `CMD` in the **Action** column and `IXC102A` in the **Message ID Description** column (or `IXC402D` for IXC402D message automation). For more information, refer to the online help or the section "More About Policy Item MESSAGES/USER DATA" in *IBM Tivoli System Automation for z/OS Defining Automation Policy*. The definitions here also apply to message IXC402D.

Pressing Enter will bring up the CMD Processing panel, as shown in Figure 19 on page 125. Use this panel to specify a valid command for the image and a "Pass/Selection" value that must match the "Value Returned" definition specified on the *Code Processing* panel.

```
ACTCODE
 LOAD P(LOADPROF) CLEAR
```

Figure 19. Sample Panel for Command Processing

On the *Code Processing* panel, as shown in Figure 20, specify the following:

```
Code 1          Code 2          Code 3          Value Returned
IXC102A         BCPII                           ACTCODE
```

Figure 20. Sample Panel for Code Processing

If you want to automate messages IXC102A and IXC402D using the Parallel Sysplex enhancements, you must enter `IXC102A` for Code 1 and `BCPII` for Code 2. Refer to "Important Processor Operations Considerations" on page 127 for more information.

Enabling Long Running Enqueues (ENQs)

If you automate long running ENQs, you must define the following:

- The resource or resources that are being checked
- The time frame when a long ENQ is detected

If you automate "hung" commands, you must define the following:

- The command (or commands) that are being monitored or excluded from monitoring
- The time frame for each command that a command is granted for completion or, if commands are to be excluded from monitoring, the exclusion keyword
- The action to be taken against this command if this command is determined to be a blocker of other commands or jobs

In addition, the following definitions can be made:

- The names of jobs that should be canceled or kept when detecting a long ENQ, a "hung" command, or command flooding
- The snapshot interval for a command class
- The title of the dump taken before the job is cancelled
- The default storage areas to be dumped
- Symbol definitions to be used when the dump specifications are provided by a PARMLIB member

Use the entry type GRP in the customization dialog to define the following policies:

- Resource definition
- JOB/ASID definitions
- IEADMC*xx* symbols
- Command definition
- Snapshot interval definition

Step 1: Defining Resources

Use the Long Running ENQ Resource Definition panel to define your resources. This panel is displayed if you select policy item RESOURCE DEFINITIONS from the Long Running Enqueue Policy section of the Policy Selection panel for sysplex

groups. For more information, refer to the online help or the section "More About Policy Item RESOURCE DEFINITIONS" in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Step 2: Making Job/ASID Definitions

Use the Long Running ENQ Job/ASID Definitions panel that is displayed if you select policy item JOB/ASID DEFINITIONS from the Long Running Enqueue Policy section of the Policy Selection panel for sysplex groups. For more information, refer to the online help or the section "More About Policy Item JOB/ASID DEFINITIONS" in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Step 3: Defining IEADMC*xx* Symbols

Use the *Long Running ENQ IEADMCxx Symbols* panel that is displayed if you select policy item IEADMCxx SYMBOLS from the Long Running Enqueue Policy section of the Policy Selection panel for sysplex groups. For more information, refer to the online help or the section "More About Policy Item IEADMCxx SYMBOLS" in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Step 4: Defining Commands

Use the Long Running Command Definition panel to define your commands. This panel is displayed if you select policy item COMMAND DEFINITIONS from the Long Running Enqueue Policy section of the Policy Selection panel for sysplex groups. For more information, refer to the online help or the section "More About Policy Item COMMAND DEFINITIONS" in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Step 5: Defining Snapshot Intervals

Use the Command Flooding Definition panel to define the individual snapshot times. This panel is displayed if you select policy item COMMAND FLOODING from the Long Running Enqueue Policy section of the Policy Selection panel for sysplex groups. For more information, refer to the online help or the section "More About Policy Item COMMAND FLOODING" in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Enabling Auxiliary Storage Shortage Recovery

To prevent auxiliary storage shortage outages you can predefine local page data sets, using the SA z/OS customization dialog for entry type GRP to define the following:

- local page data set
- job definitions

Step 1: Defining the Local Page Data Set

Use the Local Page Data Set Recovery panel that is displayed if you select policy item LOCAL PAGE DATA SET from the Local Page Data Set Policy section of the Policy Selection panel for sysplex groups. For more information, refer to the online help or the section "More About Policy Item LOCAL PAGE DATA SET" in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Step 2: Defining the Handling of Jobs

Use the Local Page Data Set Recovery Job Definition panel that is displayed if you select policy item JOB DEFINITIONS from the Local Page Data Set Policy section of the Policy Selection panel for sysplex groups. For more information, refer to the online help or the section "More About Policy Item JOB DEFINITIONS" in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Defining Common Automation Items

There are definitions that relate to utilities running as a started task. The first one (Temporary Data Set HLQ/TEMPHLQ) replaces the usage of the first qualifier of the automation status file. The second definition (Started Task Job Name/STCJOBNM) allows the unique assignment of started task job names scheduled by the automation in case you have dedicated job name assignments that conflict with the procedure names provided by the automation.

It is recommended that you define the Temporary Data Set HLQ/TEMPHLQ. If it is not defined, the automation uses the first qualifier of the automation status file.

You can define both of these items using the Sysplex Policy Definition panel that is displayed if you select the policy item SYSPLEX from the Policy Selection panel for sysplexes. For more information, refer to the online help or the section "More About Policy Item SYSPLEX" in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Important Processor Operations Considerations

Currently, the IXC102A automation and Coupling Facility activation or deactivation is the product automation that uses the BCP (Basic Control Program) Internal Interface to control the processor hardware.

If you use the automation capabilities of SA z/OS processor operations in your environment, make sure they do not conflict with the automation supplied by the Parallel Sysplex enhancements.

With the Parallel Sysplex enhancements, IXC102A and IXC402D automation uses the BCP Internal Interface, which is currently not fullycompatible with processor operations.

If you want to use the IXC102A automation that is supplied as part of the Parallel Sysplex enhancements, make sure there is no processor operations related IXC102A automation defined in your automation policy.

Likewise, if you want to continue to use the processor operations based automation of messages IXC102A and IXC402D, the IXC102A automation flag provided by the Parallel Sysplex enhancements must be disabled.

Processor operations, which is a Focal Point type function, allows you to monitor and control processor hardware, including Coupling Facility images, from a single NetView, the processor operations Focal Point.

The BCP Internal Interface of the Parallel Sysplex enhancements allows you to perform hardware operations from each NetView in your sysplex member, as long as its processor hardware supports this. Refer to *IBM Tivoli System Automation for z/OS Planning and Installation* for more information.

Customizing the System to Use the Functions

Additional Automation Operator IDs

To support the Parallel Sysplex enhancements, you must define the following automation operators:

Automation Operator ID	Automated Function	Profile
AUTXCF	XCFOPER	AOFPRFAO
AUTXCF2	XCFOPER2	AOFPRFAO
AUTPLEX	PLEXOPER	AOFPRFAO
AUTPLEX2	PLEXOPR2	AOFPRFAO
AUTPLEX3	PLEXOPR3	AOFPRFAO
AUTHW001	HWOPER01	AOFPRFAO
AUTHW002–AUTHW033	HWOPER02–HWOPER33	AOFPRFHW

After you made the definitions, you have to build the new definition files via the customization dialog build function. Recycle your automation NetViews to activate the changes in the DSIPARM members.

Note: If you have different naming conventions in your setup and you change the NetView autotask IDs in the parmlib member AOFOPFA, you have to change the Primary Automation Operator fields of the AOP definitions accordingly.

Switching Sysplex Functions On and Off

Use the SA z/OS customization dialog to specify the following minor resource names:

CDS For the recovery of alternate CDSs.

ENQ Enables the handling of the next four individual recoveries.

ENQ.CMDFLOOD
Enables the handling of commands that flood a particular command class.

ENQ.HUNGCMD
Enables the handling of jobs and commands that inhibit other commands from completing execution.

ENQ.LONGENQ
Enables the handling of long-running ENQs.

HEALTHCHK For checking active sysplex settings and definitions.

LOG For the recovery of the system log.

LOGGER For the recovery of the system logger.

PAGE For the recovery of auxiliary storage shortage.

WTO For the recovery of WTO(R) buffer shortages.

XCF For automating messages IXC102A and IXC402D.

By default, all recovery actions are enabled. If you want to disable them, use the customization dialog *Flag Automation Specification* and set the recovery flag to NO.

Note: You can change the automation recovery flag during run time by using the command INGAUTO.

Chapter 11. DB2 Automation for System Automation for z/OS

Automation has been produced to provide automated functions for the DB2 software product.

Unlike other SA z/OS automation products (CICS, IMS and OPC), DB2 Automation has been designed as part of base automation. Consequently DB2 is treated as a normal SA z/OS application, relying heavily on base functionality. Therefore the material provided here should be read in conjunction with base documentation. Only DB2 Automation-specific information is provided in this document.

Overview

Automated functions provided by DB2 Automation are implemented using two distinct methods. The first being line mode invocation which allows for an operator (or OPC) initiated task to be performed on an on-demand basis. The second method is via event-driven functions such as timer expiration and NetView automation table traps. Timed commands are mainly used to provide for connection monitoring of links to IMS and CICS. Automation table commands support the dynamic discovery of IMS and CICS connections as well as Critical Event Monitoring.

Line Mode Functions

DB2 Automation offers the following operator line command functions:

- **Maintenance Start**

  This provides the ability to start DB2 in a non-standard mode. This is a command line function that will allow for an ACCESS(MAINT) type start and/or a PARM(modname) start.
- **Terminate Threads**

  This provides the ability to stop threads attached to DB2 in order to free DB2 to perform special tasks (backup).
- **Start/Stop Tablespace**

  This provides the ability to stop or start a specific Tablespace.
- **Event-Driven Functions**

  DB2 Automation event driven functions are via Timer commands or NetView automation table (AT) Traps.
- **Connection Monitoring (Timer and AT-Driven)**

  This function will facilitate the monitoring of IMS and CICS connections. This is done at connection level via either an ACF entry definition or dynamic self-discovery, or both. If required, a recovery command will be issued to re-establish a lost connection.
- **Critical Events (AT-Driven)**

  This function is handled at 2 levels, each of which can forward messages to SDF:

  1. Specific event (for example, excessive logging)
  2. General events using a NetView AT entry to drive a message threshold/recovery process

Planning Requirements

DB2 commands are issued by the SA z/OS automation work operator (AUTWRK*nn*) that the DB2 subsystem has been assigned to. Because these operators are assigned automatically during ACF load each of the AUTWRK*nn* operators that is defined to automation should be granted SYSOPR authority to DB2.

For dynamic discovery of connections, certain messages must be available:

IMS

For IMS, connection monitoring requires that messages DSNM001I, DSNM002I and DSNM003I are available to automation in order to detect the current status of a DB2 connection with IMS. This requires that IMS automation is installed. For non-DBCTL regions the IMS automation message policy must be updated with the above-mentioned message IDs so as to expose them to automation via the IMS automation AOI user exit. For more information please refer to *IBM Tivoli System Automation for z/OS IMS Automation Programmer's Reference and Operator's Guide*. The IMS non-DBCTL regions should also be defined to SA z/OS in order for the dynamically discovered connections to have recovery commands issued as a reply to the correct subsystem.

CICS

Connection monitoring requires that messages DFHDB2023I, DFHDB2025I and DFHDB2037 are available to automation.

Defining Automation Policy

Tailoring Your DB2 ACF Entries

After you have created your policy database you will need to edit it in order to add your specific DB2 Automation requirements.

SA z/OS supplies a sample policy, *DB2, that provides best practice definitions for running DB2 Automation.

Diagrams of the sample policies are provided as PDF files that are located in the USS installation path. The default for this path is: /usr/lpp/ing/doc/policies/.

You can customize DB2 Automation for your specific installation by modifying the policy database in the customization dialog. To do this, follow these steps:

Step 1. Select option **4 Policies** from the Customization Dialog Primary Menu.

Step 2. Select the required policy database that is to contain the DB2 subsystem from the Policy Database Selection panel.

Step 3. Select entry type Application from the Entry Type Selection panel.

Step 4. From the Entry Name Selection panel for Applications, type `NEW entryname` on the command line and press Enter to create a new policy object that will represent the DB2 MSTR subsystem.

Step 5. On the Define New Entry panel, you will need to enter the DB2 master subsystem name, an application type of DB2, the subtype (one of: MSTR, SPAS, IRLM, DBM1, DIST, WLMS) and the MVS job name, where *db2id* represents the prefix that you used when you defined your DB2 job names to z/OS:

```
Subsystem Name. . . . . . . . subsystem
Application Type. . . . . . . DB2
Subtype . . . . . . . . . . . MSTR
Job Name. . . . . . . . . . . db2idMSTR
```

Step 6. Press End to save this information. This will bring you to the Policy Selection panel for Applications.

Step 7. Select the policy item LINK TO CLASS.

Note: One policy item that is inherited at this point is the SHUTDOWN NORM command. This command looks like:

```
INGRDTTH &SUBSAPPL S
```

This has the effect of notifying of, and cancelling, any outstanding threads prior to DB2 shutdown. If you would prefer that threads are not cancelled then this command should be changed to read:

```
INGRDTTH &SUBSAPPL S N
```

Step 8. From the list presented, select CLASS_DB2_MASTER and press END, returning to the Policy Selection panel.

Step 9. Select the policy item AUTOMATION INFO.

Step 10. When presented with the Application Automation Definition panel, enter the command prefix character(s) for this DB2 subsystem in the entry:

```
Command Prefix .   . cmdprfx      Console command character(s)
```

Step 11. Press End to save this information.

Step 12. For *connection monitoring* you must enter CMD and CODE entries for a CONN message to describe any connections that require a forced monitoring action at each monitoring cycle (for example, CICS V4 connection). Refer to "Connection Monitoring" on page 142 for further details on how to add these entries to the MESSAGES/USER DATA policy item:

```
MESSAGES            Define Application messages
```

If you wish to force connection status refresh at NetView restart, then add an extra parameter "Y" to the end of the command to be issued for the ACORESTART message entry, for example:

```
AFTER 00:00:10,INGRDCNM &SUBSAPPL Y
```

This has the effect of ensuring that any lost connections during a NetView outage are recovered. This option is not required if you can rely on connections being automatically re-established (that is, CICS/TS).

Step 13. Press End to save this information.

Step 14. Select the extended DB2 subsystem information policy:

```
DB2 CONTROL         Define DB2 Control entries
```

Step 15. On the DB2 control entries panel define DB2 specific subsystem information. The *db2id* should be defined to indicate the subsystem ID and the Active log data set name should be entered.

```
DB2 subsystem id . . . . . . . db2id
Active log dataset name. . . . dsname
```

Step 16. Press End to return to the Policy Selection panel.

Step 17. Press End again to return to the Entry Name Selection panel for Applications.

Step 18. Type `NEW` *`entryname`* on the command line and press Enter to create a new policy object that will represent the DB2 DBM1 subsystem. On the Define New Entry panel, AOFGDYN3, you will need to enter the subsystem name and the MVS jobname:

```
Subsystem Name. . . . . . . . subsystem
Job Name. . . . . . . . . . . db2idDBM1
```

Step 19. Press End to save this information and the Policy Selection panel will appear.

Step 20. Select the policy item LINK TO CLASS.

Step 21. From the list presented select CLASS_DB2_DATABASE and press End, returning to the Policy Selection panel.

Step 22. Select the policy item RELATIONSHIPS.

Step 23. Type `New HASPARENT` on the command line and press Enter. In the **Supporting Resource** field of the upcoming Define Relationship panel, enter the subsystem name that you gave for the *db2id*MSTR job and press Enter. For the **Condition** field enter `StartsMeAndStopsMe`. The relevant entries on the Define Relationship panel should now look like:

```
Relationship Type. . . . . . . HASPARENT
Supporting Resource. . . . . . subsystem/APL/=
Condition  . . . . . . . . . . StartsMeAndStopsMe
```

Step 24. Press End until you reach the Entry Name Selection panel for Applications.

Step 25. Type `NEW` *`entryname`* on the command line and press Enter to create a new policy object that will represent the DB2 DIST subsystem. Perform the steps described for the DBM1 subsystem accordingly now for the DIST subsystem.

Step 26. Create a new policy object that will represent the DB2 IRLM subsystem. Perform the steps described for the DBM1 subsystem accordingly now for the IRLM subsystem.

Step 27. Create a new policy object that will represent the DB2 SPAS subsystem. Perform the steps described for the DBM1 subsystem accordingly now for the SPAS subsystem.

Step 28. After the MSTR, DBM1, DIST, IRLM, and SPAS subsystems have been created, they must be linked to an application group (APG).

Step 29. The application group should be linked to the required system(s) that this DB2 subsystem is to be automated on.

Step 30. After all the relevant ACF definitions have been entered, the ACF can be created using the BUILDF command.

DB2 Automated Functions: Line Command Functions

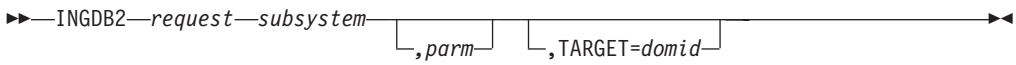
Once a DB2 subsystem has been defined to automation, there are a number of functions that can be performed against it. These are either invoked with a line command or are event-driven (timer or NetView AT).

Command Handler

Purpose

INGDB2 is the only line command delivered by DB2 Automation. This is referred to as the "Command Handler".

Syntax



Parameters

request
This can be one of the following:

START
DB2 startup (in non-standard mode), see "Maintenance Start" on page 136

TERM Terminate threads, see "Terminate Threads" on page 138

TABLE
Start or stop a tablespace, see "Start/Stop Tablespace" on page 140

subsystem
The DB2 subsystem that the request is for.

parm
A comma delimited positional parameter string.

The number of parameters depends on the command request, as follows:

START
Two parameters are available:

parm1 MAINT (for maintenance startup) or an asterisk (*, for standard start)

parm2 An optional module name for a non-standard startup.

TERM
This command request requires no parameters.

TABLE
Three parameters are available:

parm1
START (to start a tablespace) or STOP (to stop a tablespace)

parm2
A database name

parm3
A tablespace name

TARGET=*domid*
A domain in the sysplex where you want this command to be invoked (the default is the current domain). DB2 Automation must be installed on each of the domains that are required to act as a target.

Messages

```
AOF010I   WRONG NUMBER OF PARAMETERS ENTERED
AOF204I   time  : EXPECTED PARAMETERS MISSING OR INVALID FOR REQUEST clist_name
                                                                    -parameter_name
AOF332I   SUBSYSTEM name COULD NOT BE LOCATED ON target
```

Command Requests

This is a detailed description of each of the requests that can be invoked via the Command Handler.

Maintenance Start

Purpose: This function will start a DB2 subsystem in a non-standard startup mode.

Using this function, a DB2 subsystem may be started in the following non-standard startup modes:

- Maintenance mode using the default module
- Maintenance mode using a custom module
- Normal startup mode using a custom module.

If a DB2 subsystem is started in maintenance mode:

- The DDF will be stopped to inhibit any further connections
- Connection monitoring will be suppressed

To perform a standard startup (normal startup mode using default module) the SA z/OS SETSTATE command should be used to take full advantage of base automation features.

By using the INGREQ command it is possible to start DB2 with the necessary maintenance parameters entered in the "Appl Parms" field. For instance, consider an applparm field value of:

```
ACCESS(MAINT), PARM(DSNxxxxx)
```

For a normal start, this would be the equivalent of a command line maintenance start, through substitution of the &APPLPARMS parameter.

If the maintenance start parameters are consistent, they may be entered into the ACF via the flexible startup policy. This provides for a variety of start types to be identified and initiated from the INGREQ request panel. See the INGREQ command in *IBM Tivoli System Automation for z/OS Operator's Commands* for further information.

The Maintenance Start function may also be invoked from TWS Automation as a Non subsystem Operation. For further information regarding invocation from the OPC product, refer to *IBM Tivoli System Automation for z/OS TWS Automation Programmer's Reference and Operator's Guide*.

Syntax:

```
INGDB2  START subsystem,start_type,modname
```

Parameters:

subsystem
Name of the DB2 subsystem to be started in a non-standard startup mode.

start_type
Specify MAINT for a DB2 subsystem to be started in maintenance mode.

Specify * for a DB2 subsystem to be started in maintenance mode (*modname* must be specified for this option.)

modname
If *modname* is supplied then this module name will be used to start up DB2 subsystem, otherwise the default module name will be used to start up the DB2 subsystem (this parameter must be specified if *start_type* is *.

Restrictions and Limitations: Maintenance Start can only be used when:

- SA z/OS is initialized
- The DB2 subsystem is defined to SA z/OS
- A standard start is not required
- The OPC interface can only be used if TWS Automation is installed

Note: When DB2 is started in maintenance mode, the SPAS (stored procedures) address space is not started by DB2. However, SA z/OS is not aware of this and expects it to be an external startup. As a result the status of SPAS goes into an ambiguous STARTING status. The status will return to normal, when DB2 is next restarted in normal mode. For sites that frequently start up DB2 in maintenance mode, a separate DB2 application group could be defined and used in which the SPAS application is excluded.

Usage: This command may be used to start a DB2 subsystem in a non-standard startup mode, it may be invoked by the INGDB2 command handler or via TWS Automation.

Input parameters are validated for accuracy and any errors found are logged and the process is terminated. The requested subsystem is then checked to determine if automation is enabled. Once these preliminary checks have been successfully completed the requested function is initiated.

Examples: The type of startup performed will depend on the invocation parameters.

To startup a DB2 subsystem called DB2P in a non-standard startup mode, enter one of the following commands on the command line:

Example 1

```
INGDB2 START DB2P MAINT
```

Start a DB2 subsystem DB2P in maintenance mode using the default module.

Example 2

```
INGDB2 START DB2P MAINT,DSNMOD1
```

Start a DB2 subsystem DB2P in maintenance mode using custom module DSNMOD1.

Example 3

```
INGDB2 START DB2P *,DSNMOD1
```

Start a DB2 subsystem DB2P in normal mode using custom module DSNMOD1.

Policy Entries: The following DB2 startup command can be found in the sample CLASS_DB2_MASTER subsystem class STARTUP policy:

```
MVS &SUBSCMDPFX  START DB2 &APPLPARMS
```

The specified command is appended with the required invocation parameters depending on the parameters provided.

To invoke the maintenance start function via the TWS Automation interface, the following startup command is required to be coded against an "Automation Function" of UXxxxxxx.

```
INGRDMST &EHKVAR1
```

Where UXxxxxxx must match the first token of the operation text as specified in the OPC plan.

The SHUTDLY ACF entry is used to delay maintenance start should the DB2 subsystem be ACTIVE when this function is invoked.

The STRTDLY ACF entry is used to decide how long to wait before checking to see if the DB2 subsystem is UP once this function is invoked.

Messages:

```
AOF014I  SPECIFIED PARAMETER parameter INVALID
AOF146I  PARAMETER MUST BE NUMERIC
AOF204I  EXPECTED PARAMETERS MISSING OR INVALID FOR REQUEST INGRDMST - text
AOF289I  SUBSYSTEM subsystem  HAS EXCEEDED NORMAL STARTUP INTERVAL.
AOF313I  START FOR SUBSYSTEM subsystem (JOB jobname) WAS NOT ATTEMPTED - text
AOF332I  SUBSYSTEM subsystem COULD NOT BE LOCATED ON domain
AOF583I  AUTOMATION FOR SUBSYSTEM subsystem (JOB jobname) IS SET OFF -
                                                        AUTOMATION NOT ATTEMPTED
```

Return Codes:

8 Process failure—see accompanying message.

4 Automation not allowed.

0 Normal End.

-1 Command, instruction or nested command list encountered an error.

-5 Command list cancelled.

Error Codes Posted to TWS:

UX21 Automation not allowed.

UX22 Automation control file error.

UX23 DB2 subsystem cannot be started, incorrect status.

UX24 DB2 subsystem is already started.

UX25 DB2 subsystem did not start.

UX26 Error response from AOCQRY.

Terminate Threads

Purpose: The terminate threads command will terminate all active threads for a DB2 subsystem. These include REMOTE, DB2CALL, BATCH, TSO, CICS/IMS connections and all remaining ('Other') threads.

All threads are discovered using the DIS THD() command and canceled by token using CAN THD(*tokenid*). UTILITY threads are treated differently: after having been discovered using DIS UTIL() and checked not to be COPY, REORG, REPAIR or LOAD, they are terminated using the TERM UTIL() command. UTILITY threads are ignored during the cancel-threads-by-token process.

TSO users will be issued with a message informing them that their thread is about to be terminated prior to actual thread termination.

The Terminate Threads function may also be invoked from the TWS Automation as a Non-subsystem Operation. For further information regarding invocation from TWS Automation, refer to *IBM Tivoli System Automation for z/OS TWS Automation Programmer's Reference and Operator's Guide*.

Syntax:

```
INGDB2 TERM subsystem
```

Parameters:

subsystem
The name of DB2 subsystem that all active threads are to be terminated for.

Restrictions and Limitations: Terminate threads can only be used when:

- SA z/OS is initialized
- The DB2 subsystem is defined to SA z/OS
- The status of the DB2 subsystem is 'UP'
- The OPC interface can only be used if TWS Automation is installed.
- For non-utility threads, only those with a non-zero token can be canceled.

Usage: Use this command to terminate all active threads for a DB2 subsystem. It may be invoked by the INGDB2 command issued from the command line or via TWS Automation.

Input parameters are validated for accuracy and any errors found are logged and the process is terminated. The requested subsystem is then checked to determine if automation is enabled. Once these preliminary checks have been successfully completed the requested function is initiated.

Example: To terminate threads for a DB2 subsystem called DB2P, enter the following from the command line:

```
INGDB2 TERM DB2P
```

Policy Entries: DB2 Control policy item entries can be used to control the length of time to Terminate Threads.

"Terminate Threads Delay" represents the delay between each iteration of the terminate threads request.

"Cycles" represents the maximum number of iterations of the terminate threads request automation is to attempt.

To invoke the terminate threads function via the TWS Automation interface, the following startup command is required to be coded against an "Automation Function" of UXxxxxxx.

```
INGRDTTH &EHKVAR1
```

Where UXxxxxxx must match the first token of the operation text as specified in the OPC plan.

IMS BMP threads can be handled separately by requesting this via the Connection Monitoring Policy Entries. Using the CONN Message Policy entry for the DB2 subsystem, create a coded entry as required by the CDEMATCH common routine.

Code 1	Code 2	Code 3	Value Returned
IMSCONID	IMSCTLJB	STOPBMP	YES/NO

Where:

- **Code 1** is the IMS connection ID
- **Code 2** is the IMS Control Region job name
- **Code 3** is a fixed request identifier
- **Value Returned** is Yes or No, to indicate whether this IMS job's BMPs should be stopped using the IMS /STOP REG ABDUMP command

Messages:

```
AOF004I  PROCESSING FAILED FOR db2cmd
AOF014I  SPECIFIED PARAMETER cycle INVALID
AOF144I  PARAMETER parameter_name INVALID
AOF146I  PARAMETER MUST BE NUMERIC
AOF204I  EXPECTED PARAMETERS MISSING OR INVALID FOR REQUEST INGRDTTH - text
AOF332I  SUBSYSTEM subsystem COULD NOT BE LOCATED ON domain
AOF583I  AUTOMATION FOR SUBSYSTEM subsystem (JOB jobname) IS SET OFF -
                                                        AUTOMATION NOT ATTEMPTED
ING107E  INDOUBT THREADS EXIST - SHUTDOWN OF subsystem WILL NOT PROCEED
ING108I  NO THREADS LEFT IN subsystem
ING109E  thread_number THREADS COULD NOT BE TERMINATED FROM subsystem.
ING112I  YOUR TSO DB2 (subsystem) THREAD IS ABOUT TO BE TERMINATED BY AUTOMATION.
ING113I  YOUR TSO DB2 (subsystem) THREAD HAS BEEN TERMINATED BY AUTOMATION.
ING114E  jobname CANCELLED BY AUTOMATION DUE TO subsystem THREAD TERMINATION.
ING127A  THREADS FOUND AFTER LAST CYCLE OF DB2 (subsystem), FORCE SHUTDOWN.
```

Return Codes:

8 Process failure—see accompanying message.

4 Automation not allowed.

0 Normal End.

-1 Command, instruction or nested command list encountered an error.

-5 Command list cancelled.

Error Codes Posted to TWS:

UX41 Automation not allowed.

UX43 Termination of connections unsuccessful.

UX44 Error response from DB2 command.

Start/Stop Tablespace

Purpose: For DB2 Tablespace Start, the necessary Tablespace start command will be issued.

For DB2 Tablespace Stop, certain active threads that use the Tablespace will be terminated. These include REMOTE, DB2CALL, BATCH and TSO. TSO users will

be issued with a message informing them that their thread is about to be terminated prior to actual thread termination.

The Tablespace Start/Stop function may also be invoked from the TWS Automation as a Non-subsystem Operation. For further information regarding invocation from TWS Automation, refer to *IBM Tivoli System Automation for z/OS TWS Automation Programmer's Reference and Operator's Guide*.

Syntax:

```
INGDB2  TABLE subsystem,request_type,dbname,tsname
```

Parameters:

subsystem
The name of DB2 subsystem

request type
START (start Tablespace) or STOP (stop Tablespace)

dbname
Database name

tsname The name of the tablespace that is to be started or stopped

Restrictions and Limitations: Tablespace Start/Stop can only be used when:

- SA z/OS is initialized.
- The DB2 subsystem is defined to SA z/OS.
- The OPC interface can only be used if TWS Automation is installed.

Usage: Use this command to star tor stop a Tablespace for a DB2 subsystem. It can be invoked by the INGDB2 command handler or via TWS Automation.

Input parameters are validated for accuracy and any errors found are logged and the process is terminated. The requested subsystem is then checked to determine if automation is enabled. Once these preliminary checks have been successfully completed the requested function is initiated.

Examples:
Example 1. To start a Tablespace where the DB2 subsystem is DB2P, the database name is DB2PDBN and the Tablespace name is DB2PTSN, then enter the following command on the command line:

```
INGDB2  TABLE  DB2P,START,DB2PDBN,DB2PTSN
```

Example 2. To stop a Tablespace where the DB2 subsystem is DB2P, the database name is DB2PDBN and the Tablespace name is DB2PTSN, then enter the following command on the command line:

```
INGDB2  TABLE  DB2P,STOP,DB2PDBN,DB2PTSN
```

Policy Entries: DB2 Control policy item entries can be used to control the length of time to Terminate Threads.

"STOP tablespace delay" represents the delay between each iteration of stop Tablespace attempt.

TSO logoff delay represents the delay before issuing the TSO logoff message to users of the Tablespace.

To invoke the start/stop Tablespace function via the TWS Automation interface, the following startup command is required to be coded against an "Automation Function" of UXxxxxxx.

```
INGRDSTS &EHKVAR1[START|STOP]
```

Where UXxxxxxx must match the first token of the operation text as specified in the OPC plan. The STOP or START request parameter is optional and can be used if all of the required parameters cannot fit in the operation text field.

Messages:

```
AOF004I  PROCESSING FAILED FOR db2cmd
AOF144I  PARAMETER parameter_name INVALID
AOF204I  EXPECTED PARAMETERS MISSING OR INVALID FOR REQUEST INGRDSTS -  text
AOF332I  SUBSYSTEM subsystem COULD NOT BE LOCATED ON domain
AOF583I  AUTOMATION FOR SUBSYSTEM subsystem (JOB jobname) IS SET OFF -
         AUTOMATION NOT ATTEMPTED
ING109E  thread_no THREADS COULD NOT BE TERMINATED FROM subsystem
ING129E  jobname CANCELLED. TABLESPACE dbname.tsname(subsystem)
         NEEDED TO BE STOPPED.
ING130I  TABLESPACE dbname.tsname(subsystem) IS TO BE STOPPED. PLEASE STOP USING IT.
ING131I  YOU WERE CANCELLED BECAUSE TABLESPACE dbname.tsname(subsystem)
         IS TO BE STOPPED.
ING132I  thread_no THREADS CANCELLED DUE TO STOP OF TABLESPACE
         dbname.tsname(subsystem)
```

Return Codes:

8 Process failure—see accompanying message.

4 Automation not allowed.

0 Normal End.

-1 Command, instruction or nested command list encountered an error.

-5 Command list cancelled.

Error Codes Posted to TWS:

UXA1 Automation not allowed.

UXA2 Error response from DB2 command.

UXA4 Tablespace is still allocated.

Event-Driven Functions

This is a detailed description of each of the commands that can be invoked as the result of an event, either a NetView AT trap or a NetView Timer expiration.

Connection Monitoring

Purpose

Connection monitoring is designed to, where possible, dynamically discover CICS and IMS connections to a DB2 subsystem. Once discovered the status of a connection can be maintained by tracking the relevant messages generated as the connection is affected by the operating environment.

When dynamic discovery is not feasible (due to the inability of automating the relevant messages) then the connection information can be read from ACF CONN entry-type entries, and the status checked by issuing the relevant MVS commands.

In all cases, should the connection be found in the “DOWN” status then the necessary restart command will be automatically issued.

Restrictions and Limitations

Connection Monitoring can only be used when:

- SA z/OS is initialized.
- The DB2 connection is defined to SA z/OS.
- The CICS and IMS application is defined to SA z/OS.
- The CICS or IMS attachment facility is installed for the relevant subsystems.

Usage

This function is driven by a NetView Timer expiration in order to check the connections that are being monitored to be ″UP″ and, if necessary, issue a recovery command. This function can also be driven from the NetView AT trap in order to update the connection status.

Input parameters are validated for accuracy and any errors found are logged and the process is terminated. The requested subsystem is then checked to determine if automation is enabled. Once these preliminary checks have been successfully completed the requested function is initiated.

For the NetView AT event driven process the automation flag for the connection minor resource is **not** checked and, depending on the message that was trapped, the relevant CGlobal information for the particular connection will be updated.

If the process is driven as a result of a NetView Timer expiration then all known connections are checked for availability. The connections to check are identified from the ACF CONN entry-type entries, as well as the Cglobals built from the NetView AT driven process. The individual connection’s automation flags are checked to see if recovery should be considered. For each connection, if automation is ″ON″ and the connection status is “DOWN” (all ACF CONN connection entries are assumed to be “DOWN” for this purpose) then the connection is checked for its current status. If the connection is confirmed to be “DOWN” then a recovery command will be issued. For ACF entry identified connections the recovery command is issued from the ACF, otherwise a command is built from discovered information and then issued.

Policy Entries

DB2 Control policy item entries can be used to control the length of time between Connection Monitoring cycles.

″Connection monitor delay″ represents the delay between each NetView Timer expiration which will trigger the connection status checking cycle.

Connection monitoring is initially invoked to run on a timer initiated by the DB2 UP or NetView restart messages.

You can control Connection Monitoring by using the CONN and CONN.*connid* minor resource automation flags, where *connid* represents the name of a connection that requires automation to be switched off.

The connection identification entries can be entered into your automation policy using the Customization Dialog MESSAGES/USER DATA policy item for the DB2 subsystem. The Message ID should be CONN against which the CODE and CMD entries should be made:

Code 1	Code 2	Code 3	Value Returned
CICONID	CICS4A	CICS	CICS4A ENTRY

Where:

- **Code 1** is the connection ID (usually the CICS applid or IMS subsystem ID)
- **Code 2** is the CICS or IMS job name
- **Code 3** is the connection type (CICS or IMS)
- **Value Returned** is the connection description.

AT Entries

Messages DSNM001I, DSNM002I, and DSNM003I will trigger connection monitoring (INGRDCNM) to run from the NetView automation table.

Messages

```
AOF004I  PROCESSING FAILED FOR db2cmd
AOF144I  PARAMETER parameter_name INVALID
AOF204I  EXPECTED PARAMETERS MISSING OR INVALID FOR REQUEST INGRDCNM -  text
AOF205A  time : command COMMAND FAILED FOR clist_name : interval -
         WAIT TIME EXPIRED
AOF332I  SUBSYSTEM subsystem COULD NOT BE LOCATED ON domain
AOF583I  AUTOMATION FOR SUBSYSTEM subsystem (JOB jobname) IS SET OFF -
         AUTOMATION NOT ATTEMPTED
ING101A  subsystem  CONNECTION TO conn_desc (conn_id) DOWN. RECOVERY
         COMMAND ISSUED
ING102I  subsystem CONNECTION TO conn_desc (conn_id) IS UP.
```

Return Codes

8 Process failure—see accompanying message.

4 Automation not allowed.

0 Normal End.

-1 Command, instruction or nested command list encountered an error.

-5 Command list cancelled.

Critical Event Monitoring

Purpose

Critical Event Monitoring handles specific critical events that may occur during normal day-to-day running of DB2. These include:

Message	Description
DSNB250E, DSNB311I, DSNB312I, DSNB320I, DSNB321I, DSNB322I, DSNB323I, DSNB350I, DSNB351I	Recover failed dataspace (for data sharing only)
DSNB309I	Recover failed group buffer pool
DSNV086E	Unrecoverable/Recoverable DB2 abends
DSNJ002I	Switch active log data sets
DSNR004I	RESTART...UR STATUS COUNTS
DSNP007I	Data set could not be extended
DSNJ110E	Last active log data set is % full

Message	Description
DSNJ111E	All active log data sets full
DSNJ115I	Archive data set could not be allocated
DSNT500I/DSNT501I	Resource unavailable

Recovery commands that are required should be defined for the message in the automation's MESSAGES/USER DATA policy item for any DB2 subsystem that requires DB2 critical event message recovery.

If the DB2 subsystem is known to SA z/OS as an application of type DB2, event message recovery can be controlled by parameters entered via the DB2 CONTROL policy item for the subsystem.

AT statements that call the generic routine ISSUEACT or a DB2-specific routine are only created during the build process for messages that have commands defined in the automation control file. If a recovery action is only to be processed when the triggering message is issued by a subsystem of type DB2, the created automation table statement is labeled with the group name DB2. Otherwise the automation table statement is created without a label.

Created automation table statements that call the generic routine ISSUEACT are conditional and can be overwritten via automation policy item MESSAGES/USER DATA.

Restrictions and Limitations

Critical event monitoring is only done if the DB2 subsystem is defined to SA z/OS and if recovery is enabled for the invoking message ID minor resource.

Critical event monitoring with DB2-specific routines is only done if the DB2 subsystem is known to SA z/OS as an application of type DB2

For some recovery actions SYSOPR needs SYSCTRL authority.

Usage

For each of the Event Monitoring processes, the input parameters are validated for accuracy and any errors that are found are logged and the process is terminated. The requested subsystem is then checked to determine if automation is enabled for the major resource, and also that recovery is enabled for the invoking message ID minor resource. Once these preliminary checks have been successfully completed, the requested recovery action is performed.

DSNB250E, DSNB311I, DSNB312I, DSNB320I, DSNB321I, DSNB322I, DSNB323I, DSNB350I, DSNB351I: Recover Failed Dataspace:

Description: Affected dataspaces are identified by using the `DIS DB() SPACE() RESTRICT LIMIT()` command. Using the returned DSNT397I message, the status of each dataspace is checked for LPL or GRECP. If matched then the dataspace is tagged for recovery. This is achieved by issuing the `STA DB() SPACENAM() ACCESS()` command for each tagged database or dataspace. Priority is given to databases DSNDB01 and DSNDB06 if necessary.

Policy Entries: Any database or tablespace that is to be excluded from recovery should be entered using the MESSAGES/USER DATA policy item. The Message ID should be ″DATABASE″ against which the ″CODE″ entry should be made. For

example:

Code 1	Code 2	Code 3	Value Returned
Database	Tablespace		IGNORE

Refer to CDEMATCH in *IBM Tivoli System Automation for z/OS Programmer's Reference* for code matching rules.

DSNB309I: Recover Failed Group Buffer Pool:

Description: This function will stop DB2 on receipt of the DSNB309I for Group Buffer Pool GBP0. This is triggered using an AT trap that will invoke INGRDTTH to perform the INGREQ STOP command. SA z/OS will then attempt to start any other DB2 that is defined within the sysplex, based on preference values.

DSNV086E: Unrecoverable/Recoverable DB2 Abends:

Description: This function will identify specific DB2 abends as non-recoverable. This will cause the DB2 subsystem to *break* DB2. SA z/OS will then attempt to start any other DB2 that is defined within the sysplex, based on preference values. Other DB2 abends will be recoverable.

DSNJ002I: Switch Active Log Data Sets:

Description: If commands are defined in the automation policy item MESSAGES/USER DATA for this message to an application of type DB2, these commands are only issued when the triggering message is for the log data set that is specified by the "Active log data set name" in the DB2 CONTROL policy item.

If the application is not of type DB2, the defined commands are issued unconditionally.

DSNR004I: RESTART...UR STATUS COUNTS:

Description: If the message reports an INDOUBT counter greater than zero at the end of a DB2 restart process, ISSUEACT is called to issue commands that are defined in the automation policy item MESSAGES/USER DATA of the DB2 application for this message.

DSNP007I: Data Set Could Not Be Extended:

Description: The created automation table statement calls ISSUEACT with its code specifications as the parameters. The code values are extracted from the message text. The data set name is passed as the CODE1 value, the return code is passed as the CODE2 value, and the connection ID is passed as the CODE3 value. If a code match is found with the ID of the triggering message in policy item MESSAGES/USER DATA, the value that is returned is used to select and issue the related commands as defined in the automation policy.

DSNJ110E: Last Active Log Data Set Is % Full:

Description: If commands are defined in the automation policy item MESSAGES/USER DATA for this message to an application of type DB2, these commands are only issued when the message reports a percentage full figure that is equal to or greater than the critical threshold that is defined in the "Log full threshold" field of the DB2 CONTROL policy item.

If the application is not of type DB2, the defined commands are issued unconditionally.

DSNJ111E: All Active Log Data Sets Full:

Description: If commands are defined in the automation policy item MESSAGES/USER DATA for this message to an application of type DB2, these commands are only issued when the number of received messages within a time period exceeds a given threshold. The time period and the threshold can be entered via the DB2 CONTROL policy item in the ″Active log alerts″ field and the related ″Threshold″ field.

If the application is not of type DB2, the defined commands are issued unconditionally.

DSNJ115I: Archive Data Set Could Not Be Allocated:

Description: If commands are defined in the MESSAGES/USER DATA automation policy item for this message for an application of type DB2, these commands are only issued when the elapsed time since they were triggered by this message is greater than a given time interval, as specified in the **Log offload interval** field in the DB2 CONTROL policy item.

If the application is not of type DB2, the defined commands are issued unconditionally.

DSNT500I/501I: Generate DSNT500I/DSNT501I Alert:

Description: The created automation table statement calls ISSUEACT with its code specifications as the parameters. The code values are extracted from the message text. The name is passed as CODE1, the reason is passed as CODE2 and the type is passed as CODE3. If a code match is found with the ID of the triggering message in policy item MESSAGES/USER DATA, the value that is returned is used to select and issue the related commands as defined in the automation policy.

Chapter 12. WTOR Processing

When System Automation for z/OS receives WTORs (write-to-operator-with-reply requests), it either automatically replies to them, or stores them if they are to be used for recovery or to shut down the subsystem that issued them. WTORs that are stored for later use are known as outstanding WTORs.

Process Flow of WTORs

All WTORs that are issued at a system should be forwarded to NetView. Otherwise SA z/OS will not be able to process them.

From NetView V5R2 the following definition in the message revision table ensures that all WTORs are provided to NetView for automation

```
UPON (ALWAYS)
  SELECT
* Ensure all WTORs are being automated
    WHEN (WQE SUBSTR 345 C2D ^= "+0")
      REVISE("Y" AUTOMATE)
    OTHERWISE
  END
```

For earlier releases of NetView:

- The known WTORs that are to be forwarded to NetView have to be defined for automation in the MPF table
- The unknown WTORs have to be forwarded by means of an assembler exit

Incoming WTORs are processed by the NetView automation table (AT) and this triggers generic routines according to the processing purpose:

Called Generic Routine	Processing Purpose
ISSUEACT	1. Issue commands or replies (or both) that have been defined to a subsystem. 2. Store the WTOR if it has not been replied to.
ACTIVMSG, HALTMSG, TERMMSG	1. Update the status of the subsystem that issued the WTOR. 2. Issue defined commands or replies (or both). 3. Store the WTOR if it has not been replied to.
INGMON	1. Issue commands or replies (or both) that have been defined to a monitor resource. 2. Store the WTOR if it has not been replied to.
OUTREP	Store the WTOR.

The generic routines (other than OUTREP) are routed to the first active task that is defined in the AT synonym %AOFOPGSSOPER%. Thus they are usually routed to the work operator of the subsystem that issued the WTOR. This is done based on the job name that is associated with the WTOR.

Generic routines that process WTORs from subsystems that are not defined in SA z/OS or are from MVS components are routed to tasks that the WTORs have been assigned to based on their message ID.

The OUTREP generic routine is routed to the first active task that is defined in the AT synonym %AOFOPSYSOPER%.

Actions in Response to Incoming WTORs

You can use the MESSAGES/USER DATA automation policy item to define what response SA z/OS should make to incoming WTORs for applications, monitor resources and MVS components, as follows:

- Use the CMD action (possibly combined with the CODE action) to define commands that are to be issued in response to an incoming WTOR.
- Use the REP action (possibly combined with the CODE action) to define a reply that is to be made immediately in response to an incoming WTOR.
- Use the AUTO action to define the incoming WTOR as a status message that changes the status of the subsystem that issued the WTOR.

NetView automation table statements are created that call the relevant generic routine dependent on the defined actions.

If you used CODE definitions to define actions, the automation table statements that are created have to be supplemented with an OVR action to tell SA z/OS what variable information is to be extracted from the WTOR and how to pass this data as code values to the related generic routine.

WTORs that have no actions defined for them are stored by SA z/OS via OUTREP. Appropriate automation table statements are created for this purpose.

Customizing how WTORs Are Stored by SA z/OS

SA z/OS keeps track of all outstanding WTORs that have not yet been replied to and displays them via SDF or NMC.

These outstanding WTORs include:

- Permanent outstanding WTORs that are issued by applications at startup and thus provide an interface for critical operator communication and shutdown
- WTORs that no replies have been defined for in the SA z/OS automation policy
- WTORs that were issued before SA z/OS had initialized or during down time of SA z/OS

You can use the automation policy to define the severity for outstanding WTORs and a priority that allows you to distinguish between primary and secondary WTORs:

Severity

The severity of a WTOR determines the color of the WTOR in SDF and NMC. The following values can be specified for the severity:

NORMAL

Ordinary messages that do not indicate a problem.

UNUSUAL

Messages that might indicate a problem.

IMPORTANT

Messages that indicate serious problems.

IGNORE
Messages that are to be ignored by SA z/OS.

Priority
A primary WTOR is stored and can later be used for operator communication and to shut down the subsystem that issued it. In contrast, secondary WTORs are replied to immediately, or may be stored to be displayed in SDF and NMC.

This customization is done with code definitions in the MESSAGES/USER DATA policy item for a message ID of WTORS. For details see the description of the OUTREP generic routine in *IBM Tivoli System Automation for z/OS Programmer's Reference*.

Processing of Primary WTORs

To prevent SA z/OS from replying to primary WTORs as soon as they are received, the replies are *not* defined directly for the message ID of the primary WTOR. Instead, the issuing of replies to primary WTORs is invoked by other messages or executed commands. Thus the replies for primary WTORs that are to be deferred are defined for the ID of these invoking messages, or the replies to be issued are provided for a predetermined message ID. For example, the SHUTDOWN automation policy item is used to define the replies to be issued during shutdown.

The reply ID of any stored, primary WTOR to a subsystem can be used for operator communication or the shutdown of this subsystem.

If SA z/OS has to communicate with a subsystem by issuing a reply but an outstanding WTOR has not yet been stored for the subsystem, the RETRY option is used to wait for the required WTOR.

You can define multiple replies with the same pass or selection option for a message ID. These replies can be used in response to a sequence of incoming primary WTORs.

Example

Message ABC123D is issued by application ABCAPPL during startup as permanent, outstanding WTOR and SA z/OS stores it as primary WTOR for this application. During the lifetime of the application, whenever message ABC789I is issued in special situations, a reply should be issued to the permanent, outstanding WTOR ABC123D for this application. The MESSAGES/USER DATA automation policy item for message ID ABC789I of the application is used to define this reply.

When message ABC789I is issued by the application, SA z/OS retrieves the reply ID of the permanent, outstanding WTOR and issues command MVS R 117,ABC RESTORE, as shown in Figure 21 on page 152.

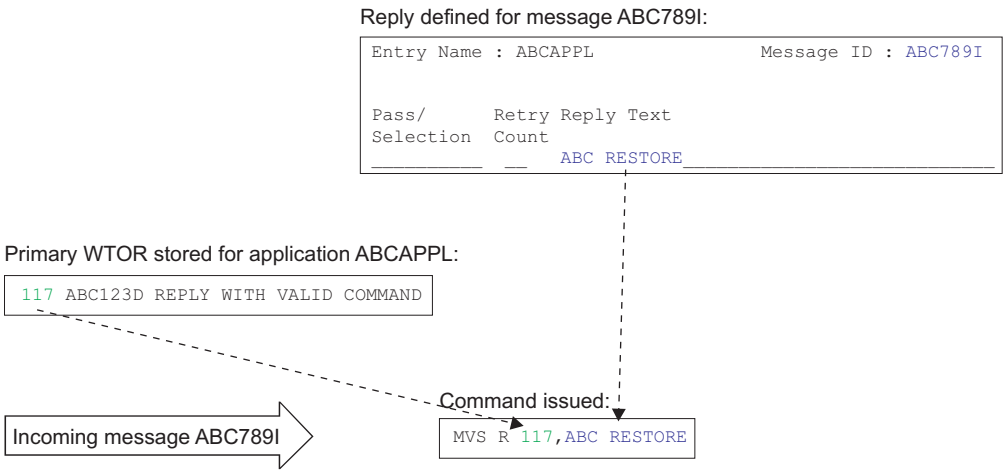


Figure 21. Example Processing of a Primary WTOR

Restrictions

The reply IDs of a subsystem's outstanding primary WTORs are stored by SA z/OS as a blank-separated list without leading zeros. The storage for this is restricted to 255 bytes. If this limit is reached, the reply IDs of further incoming primary WTORs are ignored.

Usage Notes

When storing incoming WTORs, a search for code definitions for the message ID, WTORS, is first made in the entry for the subsystem that issued the WTOR. If the subsystem itself cannot be found in the automation policy or the code definitions that are searched for are not found for the subsystem, they are searched for under the MVSESA entry. For subsystems such as IMS or NetView that have a permanent outstanding reply, you should specify the code definitions for the subsystem entries themselves instead of MVSESA. This improves performance by reducing searches within the automation policy.

Chapter 13. SA z/OS User Exits

To allow user-specific activities that are not covered by the customization dialogs, SA z/OS provides support for the following classes of user exits:

- Initialization exits that are called at the start of SA z/OS initialization, before message AOF603D is issued, see "Initialization Exits" on page 154
- Static exits that are called at fixed points during SA z/OS processing, see "Static Exits" on page 156
- Flag exits that are called when SA z/OS needs to evaluate an automation flag, see "Flag Exits" on page 159
- Customization Dialog exits that can be called during certain phases when working with the customization dialog, see "Customization Dialog Exits" on page 162
- Command exits that can be called during the processing of certain commands, see "Command Exits" on page 166

Additionally, SA z/OS has a number of facilities that behave in an exit-like manner.

Figure 22 on page 154 shows the sequence that exits may be invoked in during SA z/OS initialization.

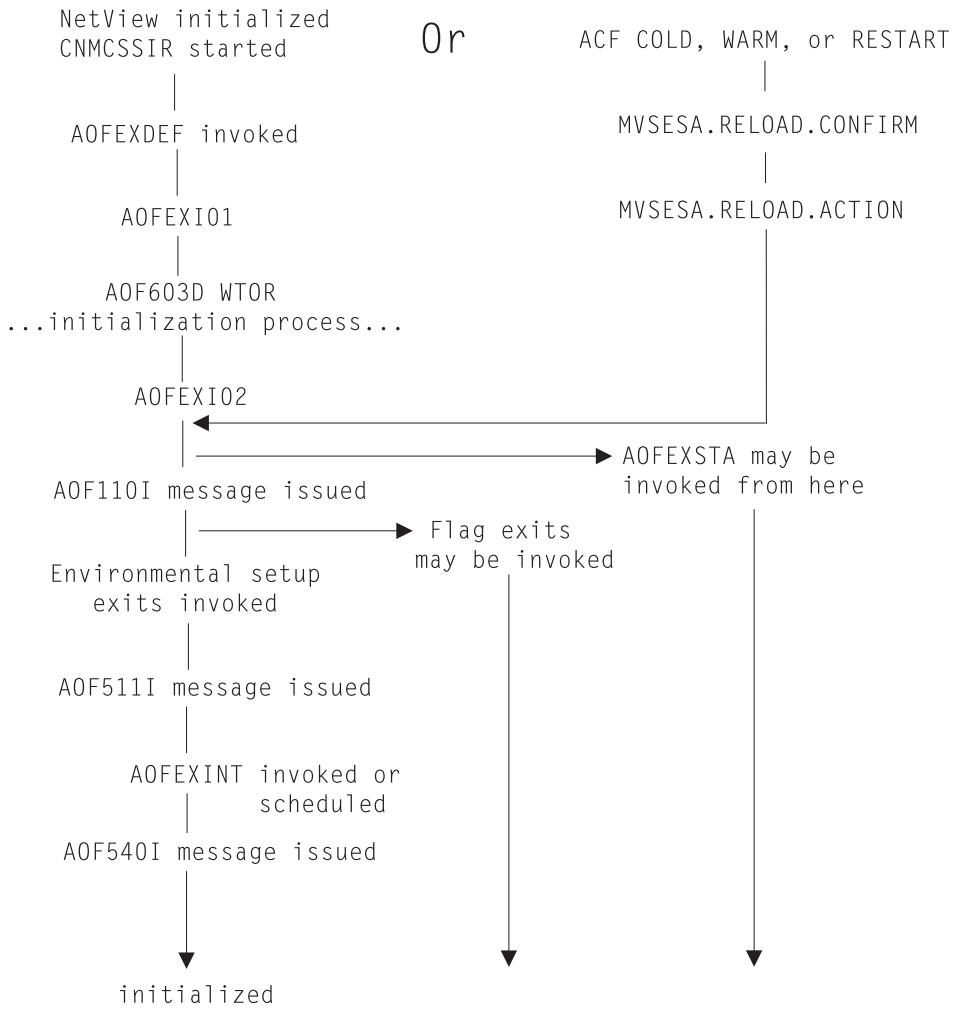


Figure 22. SA z/OS Exit Sequence during SA z/OS Initialization

Initialization Exits

These exits are invoked at the start of SA z/OS initialization, before message AOF603D is issued.

Environmental Setup Exits

The SA z/OS customization dialog allows you to define a string of exits that are invoked during SA z/OS initialization processing. These exits are defined using the SYSTEM INFO policy item of the System policy object. See *IBM Tivoli System Automation for z/OS Defining Automation Policy* for more information.

Environmental setup exits are invoked after SA z/OS has started its various tasks, but before the primary automation table has been loaded. You can use these exits to initiate your own automation, but some SA z/OS services may be unavailable because SA z/OS has not yet finished initializing when these exits are called. In particular, status information may be inaccurate because SA z/OS may not have finished resynchronization. Environmental setup exits run on AUTO1.

Parameters

Parameters are passed in sequence, delimited by blanks.

INITIALIZATION
INITIALIZATION is a constant.

Either RELOAD or REFRESH or IPL or RECYCLE

RELOAD	Indicates that the automation control file has been reloaded.
REFRESH	Indicates that the automation control file has been refreshed.
IPL	Indicates that SA z/OS has just been restarted after a system IPL.
RECYCLE	Indicates that NetView has been restarted.

Return Codes

0 is expected. If you return a non-zero return code you may prevent other exits from being invoked or disrupt SA z/OS initialization.

Usage Notes

- These exits are not driven if you run RESYNC.
- Unlike the other static exits, you must specify the name of the routine or routines to invoke in the automation control file.

AOFEXDEF

This exit is called at the start of SA z/OS initialization, before message AOF603D is issued. For example, using AOFEXDEF you can load a different MPF table.

This exit is run on AUTO1.

Parameters: None.

Return Codes: 0 is expected.

AOFEXI01

This exit is invoked before the AOF603D ENTER AUTOMATION OPTIONS reply is issued. It is invoked in a NetView PIPE and gets the data that is displayed in the AOF767I message as input in the default SAFE. With this exit you can add or remove lines from the message and add additional options to the reply.

Parameters: None.

Return Codes: 0 is expected.

AOFEXI02

This exit is invoked after the operator has replied to the AOF603D reply. It gets the operator's response to the reply as input in the default safe and it can remove, add, or change the options that the operator has entered.

Parameters: None.

Return Codes: 0 is expected.

AOFEXI03

This exit is invoked before SA z/OS loads NetView automation table. It can be used to create statistics of the currently loaded ATs. Together with the AT listings that SA z/OS produces at load, these statistics can be used for any purpose.

Parameters: None.

Return Codes: 0 is expected.

AOFEXI04

This exit is invoked after SA z/OS loads NetView automation tables. It can be used to store the AT listings that SA z/OS produces at load.

Parameters: None.

Return Codes: 0 is expected.

AOFEXI05

This exit is called before either an ACF load or refresh takes place. The parameter indicates what action the automation agent is going to process: REFRESH or LOAD.

Parameters: Type of ACF action (REFRESH or LOAD).

Return Codes: 0 (**Note:** the return code is ignored by the caller).

AOFEXI06

This exit is called after an ACF process (LOAD or REFRESH) has completed (AOFCOMPL=YES) and before the AOF540I message is issued.

Parameters: Type of ACF action (REFRESH or LOAD).

Return Codes: 0 (**Note:** the return code is ignored by the caller).

AOFEXINT

This exit is called when SA z/OS initialization is complete, before message AOF540I is issued. You can use AOFEXINT to call your own initialization processing after SA z/OS has finished. Refer also to the description of the global variable AOFSERXINT in "AOFSERXINT global variable" on page 241.

Parameters: The input parameter is the *Starttype* which is one of the following: RESYNC, IPL, REFRESH, RELOAD, RECYCLE.

Return Codes: 0 is expected.

Static Exits

These exits are invoked at fixed points in SA z/OS processing. They are always invoked if they are found in the DSICLD concatenation. Positive return codes from these exits are generally ignored, though it is recommended that you always exit with a return code of `0`.

The main purpose of static exits is to allow you to take your own actions at specific points during SA z/OS processing. The static exits available are described below.

AOFEXSTA

This exit is called from AOCUPDT every time the automation status of an application is updated.

Note: It is not necessary for AOCUPDT to *change* an application automation status for this exit to be called. The exit is still invoked if the update does not result in a change of status.

AOFEXSTA can be used to perform any special status transition processing that cannot be triggered by other methods.

Note: This exit is invoked frequently, and will be invoked at times when SA z/OS is not fully initialized. Your exit code should be as robust and efficient as possible.

SA z/OS will attempt to load AOFEXSTA into storage at initialization. If this attempt fails, AOFEXSTA will not be invoked on any AOCUPDT calls. To activate the exit it must be present in the DSICLD concatenation when the automation control file is loaded or reloaded.

AOFEXSTA runs on the task that called AOCUPDT, after all other processing has finished.

Attention: AOFEXSTA is scheduled with EXCMD opid(). If your operators are issuing commands which change application statuses and you wish to use AOFEXSTA, you may have to modify your command authorization definitions.

Parameters: Parameters are passed in sequence, delimited by commas.

Resource type
SA z/OS uses types of SUBSYSTEM, MVSESA, WTORS, and SPOOL. Other users may use other resource types.

Resource Name
For an application, this is the name of the subsystem it is defined as.

Automation Status
For an application, this is one of the automation statuses that is supported by SA z/OS.

SDF Root
This is the SDF Root, as specified in the customization dialog, for the system that originated the status update. Generally the exit is driven only for status changes on other systems on the automation focal point.

Return Codes: 0 is expected.

Restrictions:

- Because the exit is scheduled with EXCMD, the status update and subsequent processing in the caller will have completed before the exit is invoked.
- Check the resource type and the SDF root to ensure you are only trying to process the right things.
- Plan carefully before you take any action to change the status of an application from this exit. If you are not careful you may create a loop (AOCUPDT to AOFEXSTA to AOCUPDT to AOFEXSTA).

Note:

Consider using ISSUEACT or status change commands as alternatives to AOFEXSTA, because AOFEXSTA is invoked for **every** status update that seriously degrades performance.

The generic routines ACTIVMSG and TERMMSG will, if the advanced automation options are set up appropriately, issue commands whenever an application changes to a particular status. It may be more appropriate to place commands here, rather than in the status change exit, which gets driven for every status update of every resource. It is recommended to use status change commands for better performance.

AOFEXX02

The exit allows the installation to decide whether or not an SDF update should be performed for the specified resource.

A non-zero return code from the exit causes the SDF update processing to be skipped, both locally as well as for the focal point.

This exit is called prior to posting entries to SDF to provide the facility to filter out specific events.

Refer to the sample exit for details of the parameters passed to the exit and the return codes.

AOFEXX03

The exit allows the installation to decide whether or not status change notification should be forwarded to the NMC focal point for the specified resource.

A non-zero return code from the exit causes status change forwarding to be skipped.

This exit is called prior to posting entries to NMC to provide the facility to filter out specific events.

Refer to the sample exit for details of the parameters passed to the exit and the return codes.

AOFEXX04

This exit is called from CHKTHRES every time that this routine is called to check the number of errors recorded in the automation status file for a given resource against error thresholds that are defined in the automation control file.

Refer to the sample exit for details of the parameters passed to the exit and the return codes.

AOFEXX15

This exit allows you to write a log entry for each status change notification that arrives at the NMC focal point.

Refer to the sample exit for details of the parameters passed to the exit.

AOFEXX16

This exit allows you to repair your own minor resources following a topology resynchronization or rebuild.

No parameters are passed to this exit.

Flag Exits

Using automation flag exits you can cause your automated operations code to exit normal SA z/OS processing to an external source, such as a scheduling function, to determine whether automation should be on or off for a given resource at that particular instant.

Flag exits can be defined for:

- Any flag (AUTOMATION, INITSTART, START, RECOVERY, TERMINATE or RESTART).
- Any resource.
- Any minor resource. See the description of the MINOR RESOURCES policy item in *IBM Tivoli System Automation for z/OS Defining Automation Policy* for more information on minor resources.

You can specify multiple exits for each flag.

A flag exit is invoked only if SA z/OS checks the value of the current flag setting during the flag evaluation process of AOCQRY, as described in *IBM Tivoli System Automation for z/OS Programmer's Reference*. If one of the global or specific flags, which have to be checked in one iteration step during the evaluation process over the inheritance hierarchy levels is set to NO, the other flag no longer has to be checked.

With the default BYPASS option of AOCQRY, exits that have been defined for the automation flag of a resource are executed when that automation flag is checked during flag evaluation and the flag value is EXITS.

With the FORCED option of AOCQRY, exits that have been defined for the automation flag of a resource are executed when that automation flag is checked during flag evaluation, independent of the flag value, as long as it is not empty.

If an automation flag is set to EXITS, the flag value is assumed to be YES during flag checking as long as none of the exits that have been defined for the checked resource switch the flag to NO. Exits that are forced to execute do not change the flag value.

Flag settings are determined by:

- The automation policy settings
- NOAUTO periods (the flag is OFF during a NOAUTO period)
- User-entered INGAUTO command

For example, if the following flag settings are entered:

```
Resource     Flag         Setting
-----------  ------------ --------------------------
DEFAULTS     AUTOMATION   ON
SUBSYSTEM    RESTART      OFF
```

```
JES2          AUTOMATION   Call Exit J2AUT
JES2          START        Call Exit J2STR
JES2          TERMINATE    Call Exits J2SD1 and J2SD2
JES2          RECOVERY     OFF
```

The effective flags for JES2 are:

```
Flag          Effective setting
------------  ------------------------------------
AUTOMATION    Call Exit J2AUT
INITSTART     ON
START         Call Exit J2STR
RECOVERY      OFF
TERMINATE     Call Exits J2SD1 and J2SD2
RESTART       OFF
```

When SA z/OS checks the current value of any flag for the JES2 application, the process is as follows:

Flag	Process
AUTOMATION	1. Call exit J2AUT. 2. If the exit returns: • **OFF:** AUTOMATION flag is OFF • **ON:** AUTOMATION flag is ON
INITSTART	1. Call exit J2AUT. 2. If the exit returns: • **OFF:** INITSTART flag is OFF • **ON:** INITSTART flag is ON
START	1. Call exit J2AUT. 2. If exit returns ON, call exit J2STR. 3. If: • **Both flags are ON:** START flag is ON • **Either flag is OFF:** START flag is OFF
RECOVERY	RECOVERY flag is OFF
TERMINATE	1. Call exit J2AUT. 2. If exit returns ON, call exit J2SD1. 3. If exit J2SD1 returns ON, call exit J2SD2. 4. If: • **Both flags are ON:** TERMINATE flag is ON • **Either flag is OFF:** TERMINATE flag is OFF
RESTART	1. Call exit J2AUT. 2. Because either flag is OFF, RESTART flag is OFF.

Note: Normally START and RECOVERY flags are only checked by SA z/OS for minor resources but not for the subsystem itself.

Parameters

Parameters are supplied in sequence, delimited by blanks.

Flag
This is the name of the flag that is being checked. Possible values are AUTOMATION, INITSTART, START, RECOVERY, TERMINATE or RESTART.

Time Setting
Time Setting is a constant. It can be either:

AUTO
Automation is currently turned on.

NOAUTO
Automation is currently turned off.

A value of NOAUTO is possible only if AOCQRY is called with the parameter EXITS=FORCED.

Note: This ensures that the exit is invoked, but it is not possible for an exit to override a NOAUTO period.

Resource Name
This is the name of the resource that the flag is being requested for. For minor resources it will contain the fully-qualified minor resource name. Given no flag definition for TSO.USER.MAG1 and an exit enabled for TSO.USER, the resource name passed to the exit would be TSO.USER.MAG1 if a check was made for TSO.USER.MAG1.

Resource Type
This is the type of the resource that the flag is being requested for. Possible values are DEFAULTS, SUBSYSTEM, or MVSESA (the value of the common global variable AOFSYSTEM).

Target Prefix
This is the TGPFX value with which AOCQRY was invoked. If TGPFX is not specified, the value SUB is passed.

Task Global Variables

The task global variables that are set by the common routine AOCQRY are available in flag exits.

Return Codes

0 Automation is allowed by the exit.

> 0 Automation is not allowed.

Notes:

1. Flag exits are always called through common routine AOCQRY. This means that the task global variables for the resource have been primed and are available for use. Normally the names of the task global variables are prefixed with SUB, but if AOCQRY is called with a different value for parameter TGPFX, they will be found in variables that are prefixed with that value. You should use the TGPFX parameter that is passed to locate the task global variables.
2. The set of task global variables that are set by AOCQRY depends on the values for the resource and request parameter. Make sure that the task global variables that you rely on in your exit are being set up.
3. If an exit is invoked for a minor resource, the task global variables are set for the major resource that is associated with that minor resource.
4. If you call AOCQRY from within your exit you must specify a TGPFX value that is different from the TGPFX parameter value you were passed. You are responsible for ensuring the uniqueness of all TGPFXs if you nest AOCQRY exits. Because this can become quite complex, it is recommended you avoid nesting exits.

5. Do not code calls to ACFCMD, ACFREP, or CDEMATCH because these use task global variables that are prefixed with SUB that may not be set up for the application that you want to process.
6. Do not change any of the task global variables that have been set by AOCQRY.
7. Flag exits may be called frequently, so performance is important.
8. If AOCQRY is specified with FORCE and multiple exits are defined for a flag, the exits are called in order.

Customization Dialog Exits

SA z/OS provides a series of user exits that can be invoked during certain phases while working with the customization dialog. They are:

- "User Exits for BUILD Processing"
- "User Exits for COPY Processing" on page 163
- "User Exits for DELETE Processing" on page 164
- "User Exits for CONVERT Processing" on page 164
- "User Exits for MIGRATION, RENAME, and IMPORT Functions" on page 165

"Invocation of Customization Dialog Exits" on page 165 provides information on how to activate the user exits.

User Exits for BUILD Processing

The following user exits are provided for the process of building the automation control file (BUILDF).

- INGEX10, which is called before the automation control file build function starts. This exit is only available when the build process is initiated from the customization dialogs.
- INGEX01, which is called before the automation control file build function starts. starts. This exit is available when the build process is initiated from the customization dialogs, from a batch job submitted via the customization dialogs, or from a batch job submitted independently from the customization dialogs.

  When a BUILD mode of BATCH is selected in the customization dialogs, the JCL for the batch job is submitted and INGEX01 is called when the job begins execution and before the automation control file build function starts in batch.
- INGEX02, which is called after the automation control file build function (BUILDF) has ended. This exit is available when the BUILD process is initiated from the customization dialogs, from a batch job submitted through the customization dialogs, or from a batch job submitted independently from the customization dialogs.

The following parameters are passed to both INGEX01 and INGEX02 exits, separated by commas:

- Parm1 = PolicyDB name
- Parm2 = Enterprise name
- Parm3 = BUILD output data set
- Parm4 = entry type (or blank)
- Parm5 = entry name (or blank)
- Parm6 = BUILD type (MOD/ALL)
- Parm7 = BUILD mode (ONLINE/BATCH)
- Parm8 = Configuration (0=NORMAL/1=ALTERNATE)

- Parm9 = Sysplex name (or blank)
- Parm10 = Build option (1,2, or 3)
- Parm11 = return code (for INGEX02 *only*)

If user exit INGEX10 produces return code RC = 0, BUILDF processing continues. If a return code RC > 0 is produced, an error message is returned and the BUILDF processing terminates.

If user exit INGEX10 ends with return code RC > 0, user exits INGEX01 and INGEX02 are not called. Processing terminates.

If user exit INGEX10 ends with return code RC > 0 and a BUILD mode of BATCH was selected in the customization dialogs, no JCL is submitted to run the build in batch (because BUILDF does not start). Processing terminates.

If user exit INGEX01 produces return code RC = 0, BUILD processing continues. If a return code RC > 0 is produced, an error message is returned. BUILDF processing terminates. If the build is run in batch mode, and a return code RC > 0 is produced, the job finishes with a return code RC 08.

If user exit INGEX01 ended with return code RC > 0, user exit INGEX02 are not (because BUILDF does not start). Processing terminates.

User exit INGEX02 is always called when the BUILD process has started, irrespective of whether it has completed or not.

If user exit INGEX02 produces a return code RC > 0, an error message is displayed. If the build is run in batch mode, and a return code RC > 0 is produced, the job completes with a return code RC 04. If a severe build error occurred, the job completes with a return code RC 20.

The return codes and their meaning are as follows:

0 Successful

4 Build with minor errors

12 No build (data is inconsistent)

20 No build (severe errors)

User Exits for COPY Processing

Two user exits are implemented for the COPY processing.

1. INGEX03, which is called before the COPY function starts. The following parameters are passed:
   - Entry name of the entry to be copied to (target)
   - Entry name of the entry to be copied from (source)
   - Entry type (e.g. APL)
2. INGEX04, which is called after the COPY function has ended. The following parameters are passed:
   - Entry name of the entry to be copied to (target)
   - Entry name of the entry to be copied from (source)
   - Entry type (e.g. APL)
   - Indicator whether the COPY process was successful or not (S=successful, U=unsuccessful)

If user exit INGEX03 produces return code RC = 0, the COPY processing continues. If a return code RC > 0 is produced, an error message is displayed, the COPY function will not start, and processing terminates.

If user exit INGEX03 ended with return code RC > 0, the user exit INGEX04 will not be called as the COPY processing will terminate.

User exit INGEX04 is always called once the COPY function has started. The information about the success or failure of the COPY function is passed as a parameter.

If user exit INGEX04 produces a return code RC > 0, an error message is displayed.

User Exits for DELETE Processing

Two user exits are implemented for the DELETE processing.

1. INGEX05, which is called before the DELETE process starts. The following parameters are passed:
   - Entry name of the entry to be deleted
   - Entry type (e.g. APL)
2. INGEX06, which is called after the DELETE process has ended. The following parameters are passed:
   - Entry name of the entry to be deleted
   - Entry type (e.g. APL)
   - Indicator whether the DELETE process was successful or not (S=successful, U=unsuccessful)

If user exit INGEX05 produces return code RC = 0, the DELETE processing continues. If a return code RC > 0 is produced, an error message is displayed, the DELETE function will not start and the processing terminates.

If user exit INGEX05 ended with a return code RC > 0, user exit INGEX06 will not be called as the DELETE processing will terminate.

User exit INGEX06 will always be called once the DELETE function has started. The information about the success or failure of the DELETE function will be passed as a parameter.

If user exit INGEX06 produces a return code RC > 0, an error message will be displayed.

User Exits for CONVERT Processing

Two user exits are implemented for the CONVERT processing.

1. INGEX07, which is called before the CONVERT process starts. No parameters are passed.
2. INGEX08, which is called after the CONVERT process has ended. No parameters are passed.

If user exit INGEX07 produces return code RC = 0, the CONVERT processing continues. If a return code RC > 0 is produced, an error message is displayed, the CONVERT function will not start and the processing terminates.

If user exit INGEX07 ended with a return code RC > 0, user exit INGEX08 will not be called as the CONVERT processing will terminate.

User exit INGEX08 will always be called once the CONVERT function has started.

If user exit INGEX08 produces a return code RC > 0, an error message will be displayed.

User Exits for MIGRATION, RENAME, and IMPORT Functions

The following user exits are provided for the migration, renaming, and import functions.

1. INGEX09: Called when the log data set is switched, usually because the current data set is full. One parameter is passed:
   - Name of current log data set, for example, the data set that went out of space
2. INGEX12: Called after the MIGRATION function has ended. The following parameters are passed:
   - MIGRATE mode (ONLINE or BATCH)
   - Target system entry name
   - Source data set name with member (enclosed in quotes)
3. INGEX14: Called after an entry has been deleted while the MIGRATION function is running. The following parameters are passed:
   - Entry Name
   - Entry Type
4. INGEX15: Called before an entry is renamed. The following parameters are passed:
   - Entry Name
   - Entry Type
5. INGEX16: Called after an entry has been renamed. The following parameters are passed:
   - Entry Type
   - Old Entry Name
   - New Entry Name
6. INGEX17: Called during the IMPORT function, when reading data from the source policy database. One parameter is passed:
   - Name of copy data work table. This table contains the entry types and entry names of the data to be copied.
7. INGEX18: Called after the IMPORT function has ended. INGEX18 is only called if INGEX17 was called at the beginning of the IMPORT function. If checks have been made that prevent INGEX17 being called, INGEX18 is not called either.

   One parameter is passed:
   - Indicator whether the IMPORT process was successful (S=successful, U=unsuccessful)
8. INGEX20: Called after the links have been changed. No parameters are passed.
9. INGEX21: Called before the policy database report is invoked. No parameters are passed.

Invocation of Customization Dialog Exits

The user exits are part of the SA z/OS product. Therefore they are supplied in the same data set as all other ISPF REXX modules (part of SINGIREX). The supplied samples for ACF BUILD, DELETE, and COPY processing just do a 'RETURN' with return code RC=0.

You have two possibilities to apply your user modifications:

1. Edit the user exit(s) in the supplied library. Your changes will not have any consequences on the code of the SA z/OS, product. These exits will not be serviced (via PTF) by IBM as they do not include any code at the time of product delivery.
2. Supply the modified user exit in a private data set. Then you have to concatenate your private data set to your SYSEXEC library chain. As INGDLG supports multiple data set names specified for ddname SYSEXEC, this can be done in the following way:

   ```
   INGDLG SELECT(ADMIN) ALLOCATE(YES) HLQ(SYS1)
          SYSEXEC(usr.private.dsn SYS1.SINGIREX)
   ```

   This example assumes that the high level qualifier of the data sets where the IBM supplied parts exist is SYS1.

   If you specify the SYSEXEC parameter in the INGDLG call, you need to specify the IBM supplied library explicitly with its **fully qualified** data set name.

Command Exits

These exits can be called during the processing of certain commands.

AOFEXC00

The AOFEXC00 exit routine will be called if the selection L has been entered in the AOC command dialog. No parameters are passed to the routine. The purpose of this routine is to act as the starting point for installation-provided local functions.

AOFEXC01

If this exit is defined, it will be invoked during INGREQ processing before Precheck and Verification processing.

The exit allows you to modify the parameters that are passed.

Refer to the sample exit for details of the parameters that are passed to the exit and the return codes.

AOFEXC02

If this exit routine is defined, it is invoked during INGSCHED processing before the schedule override file is updated. The parameters are positional and separated by a comma.

Refer to the sample exit for details of the parameters that are passed to the exit and the return codes.

AOFEXC03

If this exit routine is defined, it is invoked by the DISPINFO command slave to retrieve user-supplied information about the subsystem. The input for the routine is the subsystem name. The data returned by the exit is shown as part of the DISPINFO output.

Refer to the sample exit for details of the parameters that are passed to the exit and the return codes.

AOFEXC04

If this exit routine is defined, the command code U is supported for the DISPMTR, DISPSTAT, and INGLIST commands. The input for the AOFEXC04 exit is the resource name (subsystem name for DISPSTAT) and the location of the resource. The location is either the system name if the resource resides on a system member of the local sysplex, or the domain ID if the resource resides on a system which is outside of the local sysplex. The parameters are separated by a comma.

Refer to the sample exit for details of the parameters that are passed to the exit and the return codes.

AOFEXC05

This exist is called on entry of the INGLIST command. The exit allows you to modify the input parameters. The modified input parameters are returned to the INGLIST command by sending a message (single or multiline) to the console, for example:

```
OBSERVED=* DESIRED=*
```

Refer to the sample exit for details of the parameters that are passed to the exit and the return codes.

AOFEXC06

This exist is called on entry of the INGSET command. The exit allows you to perform authorization checking of the resources for the INGSET command.

Refer to the sample exit for details of the parameters that are passed to the exit and the return codes.

AOFEXC07

This exist is called on entry of the INGIMS command. The exit allows you to perform authorization checking of the IMS subsystem that is the subject of the INGIMS command.

Refer to the sample exit for details of the parameters that are passed to the exit and the return codes.

AOFEXC08

This exit is called on entry of the INGVOTE command. The exit allows you to perform authorization checking of the resources for the INGVOTE command.

Refer to the sample exit for details of the parameters that are passed to the exit and the return codes.

AOFEXC09

This exit is called on entry of the SETSTATE command. The exit allows you to perform authorization checking of the resources for the SETSTATE command.

Refer to the sample exit for details of the parameters that are passed to the exit and the return codes.

AOFEXC10

This exit is called on entry of the INGEVENT command. The exit allows you to perform authorization checking of the resources for the INGEVENT command.

Refer to the sample exit for details of the parameters that are passed to the exit and the return codes.

AOFEXC11

This exit is called on entry of the INGCICS command. The exit allows you to perform authorization checking of the resources for the INGCICS command.

Refer to the sample exit for details of the parameters that are passed to the exit and the return codes.

AOFEXC12

This exit is called on entry to the command slave (EVJRVCMD) for the TWS command server (EVJRVCM0). The exit allows you to perform authorization checking of the commands scheduled via the TWS batch interface (EVJRYCMD) against the user ID of the batch job requesting the command.

Refer to the sample exit for details of the parameters passed to the exit and the return codes.

AOFEXC13

This exit is called on entry to the INGGROUP and INGMOVE commands. The exit allows you to perform authorization checking of the user ID that issues the command.

Refer to the sample exit for details of the parameters passed to the exit and the return codes.

AOFEXC14

This exit is called by the SA GDPS® termination routine (INGRGDPS) after stopping the PAM or selecting a SAM to become the PAM.

Refer to the sample exit for details of the return codes.

AOFEXC15

If this exit routine is defined, it will be invoked during INGREQ processing after the GO confirmation has been received.

The user exit is called in a PIPE. Refer to the sample exit for details of the parameters that are passed to the exit.

AOFEXC17

This exit is invoked by the INGALERT command. The exit serves two purposes:

- To modify the alert message. The first message returned by the exit is taken as alert message.
- To tell SA z/OS to skip sending the alert to System Automation for Integrated Operations Management. This is done by returning with a non-zero return code.

Refer to the sample exit for details of the parameters that are passed to the exit and the return codes.

AOFEXC20

This exit is called when a command is passed via the TWS request interface. The exit allows the installation to perform authorization checking. The exit allows the user to modify the command that is passed. The modified command must be returned by sending a message (single-line) to the console.

The installation exit is called in a PIPE. If the exit returns a bad return code and additional data is written to the console, this data is written in the netlog. If no additional data is passed in the exit, message AOF227I is issued.

Refer to the sample exit for details of the parameters that are passed to the exit and the return codes.

Pseudo-Exits

This section discusses a number of places where SA z/OS either makes special use of a flag exit or has a function with certain, exit-like, qualities.

Automation Control File Reload Permission Exit

When an operator asks SA z/OS to reload the automation control file, SA z/OS checks the global AUTOMATION flag of minor resource MVSESA.RELOAD.CONFIRM. If the flag is set to NO, the automation control file reload is not allowed. If the flag is set to YES, the task global AOFCONFIRM is checked. If AOFCONFIRM has been set to a non-null value, the user is prompted to confirm that they want the automation control file to be reloaded.

Notes:

1. Note that an exit can be associated with the global AUTOMATION flag for this resource.
2. An automation control file cannot be loaded if the global AUTOMATION flag for the major resource MVSESA is set to N. If the global AUTOMATION flag for the minor resource MVSESA.RELOAD.CONFIRM is set to Y, reloading the ACF is permitted.

Automation Control File Reload Action Exit

After the automation control file reload permission exit is checked, when SA z/OS is committed to reloading the automation control file, it will check the global AUTOMATION flagfor minor resource MVSESA.RELOAD.ACTION. The actual setting of this flag (ON or OFF) is ignored, but any exits defined for it are invoked. All exits should return a return code of 0.

Subsystem Up at Initialization Commands

Using the customization dialog you can specify commands that are run if SA z/OS finishes resynchronizing statuses and an application is found to be up. These commands can be useful for synchronizing local automation that has been built on top of SA z/OS.

Testing Exits

Exits should be well tested with a variety of different input parameters before they are put into production. For exits that need AOCQRY task globals, you can call AOCQRY to set up the globals without evaluating the flag exits, and then invoke the exit on its own for testing purposes. This method saves the overhead of calling AOCQRY every time you run the exit.

Attention:

If you have a syntax error or a no-value-condition in your exit it can cause parts of SA z/OS to abend, resulting in severe disruption of your automation.

Chapter 14. Automation Routines

System Automation for z/OS provides automation routines that enable automatic processing of z/OS components, data sets and job scheduling systems as well as automation procedures that are useful tools in the automation processing context. By using these prefabricated automation procedures you can save the time to develop your own procedures to handle the processing in corresponding situations.

In particular these automation routines provide solutions for:

The solutions for automatic processing of these situations include definitions in the automation configuration files and automation procedures.

It is common to all the solutions that are provided that the automation procedures first determine whether automation is allowed by checking the corresponding automation flags with common routine AOCQRY. See *IBM Tivoli System Automation for z/OS Defining Automation Policy* for further information concerning types and settings of automation flags. Use the DISPFLGS command to display or temporarily change the current settings of the automation flags.

Some of the automation routines respond to messages by issuing commands from the ACF. Most of these automation routines keep track of the reception of these messages and compare the frequency of the incoming messages with predefined thresholds of infrequent, frequent and critical level. If such a defined threshold is exceeded, it is taken as option for selecting the appropriate commands according to the first field in the command entry of policy item MESSAGES/USER DATA of the ACF. If no threshold is exceeded the commands to selection option ALWAYS are issued. See "How SA z/OS Uses Error Thresholds" in *IBM Tivoli System Automation for z/OS Defining Automation Policy* for further information on setting up thresholds.

This chapter describes the details of the automation functions that are provided with SA z/OS.

LOGREC Data Set Processing

The logrec recovery function responds to system messages that indicate that the logrec data set is full or nearly full. The recovery function issues predefined commands to dump and clear the logrec data sets. While the recovery function is in progress, it prevents the automation processing being started a second time.

The logrec recovery function includes the following items:

- Automation routines AOFRSA01 and AOFRSA02, see "AOFRSA01" on page 176 and "AOFRSA02" on page 178
- Automation table entries for system messages IFB040I, IFB060E, IFB080E, IFB081I, and IFC001I
- Error threshold definitions for MVS component LOGREC
- Command specification in the MESSAGES/USER DATA automation policy item for the MVSESA/LOGREC entry/type pair in the automation control file

SMF Data Set Processing

The SMF recovery function that is provided responds to system messages that indicate that the SMF data set is full or has been switched. Predefined commands from the ACF are selected to dump and clear the contents of the SMF data set. The commands to be selected can be defined depending on the occurrence of the incoming messages. The SMF recovery function includes the following items:

- Automation routine AOFRSA03, see "AOFRSA03" on page 180
- Automation table entries for system messages IEE362A, IEE362I, IEE391A and IEE392I
- Error threshold definitions for MVS component SMFDUMP
- Command specification in the MESSAGES/USER DATA automation policy item for the MVSESA/SMFDUMP entry/type pair of the automation control file

SYSLOG Processing

The syslog function that is provided responds to messages that are queued to the syslog. The function starts an external writer to save the syslog that was queued. The commands to be selected can be defined depending on the occurrence of the incoming messages.

The syslog function includes the following items:

- Automation routine AOFRSA08, see "AOFRSA08" on page 183
- Automation table entry for system message IEE043I
- Error threshold definitions for MVS component SYSLOG
- Command specification in the MESSAGES/USER DATA automation policy item for the MVSESA/SYSLOG entry/type pair of the automation control file

System Log Failure Recovery

The system log failure recovery function that is provided responds to a system log inactive message by restarting the system log. If the system log should be available to be used as the hardcopy medium, the recovery function assigns the system log as the hardcopy medium.

The recovery commands are only issued if the occurrence of the system log inactive message that is received does not exceed a defined critical threshold.

The system log failure recovery function that is provided includes the following items:

- Automation routine INGRX740, see "INGRX740" on page 232
- Automation table entries for system messages IEE037D, IEE041I, IEE533E, IEE769E, IEE043I
- Recovery automation flag for the MVS component minor resource LOG
- Error threshold definitions for the MVS component minor resource LOG
- Command specification in the MESSAGES/USERDATA automation policy item for the MVSESA/LOG entry/type pair of the automation control file

SVC Dump Processing

The SVC dump processing function that is provided responds to an SVC dump-taken message by issuing predefined commands from the ACF to handle the dump. The commands to be selected can be defined depending on the occurrence of the incoming messages.

The provided SVC dump processing function includes the following items:

- Automation routine AOFRSA0C, see "AOFRSA0C" on page 185
- Automation table entries for system messages IEA611I and IEA911E
- Error threshold definitions for MVS component MVSDUMP
- Command specification in the MESSAGES/USER DATA automation policy item forthe following entry/type pairs of the automation control file:
  - MVSESA/MVSDUMP
  - MVSESA/MVSDUMPTAKEN
  - MVSESA/MVSDUMPRESET

Deletion of Processed WTORs from the Display

The WTOR processing function that is provided deletes WTORs from SA z/OS display capabilities when they are replied to or canceled.

The WTOR processing function includes the following items:

- Automation routine AOFRSA0E, see "AOFRSA0E" on page 189
- Automation table entries for system messages IEE400I and IEE600I

AMRF Buffer Shortage Processing

The AMRF buffer shortage processing function that is provided responds to messages that report buffer shortage of the action message retention facility (AMRF). The function issues commands from the ACF to process buffer shortage automation.

The AMRF buffer shortage processing function that is provided includes the following items:

- Automation routine AOFRSA0G, see "AOFRSA0G" on page 190
- Automation table entries for system messages IEA359E, IEA360A and IEA361I
- Command specification in the MESSAGES/USER DATA automation policy item for the following entry/type pairs of the automation control file:
  - MVSESA/AMRFSHORT
  - MVSESA/AMRFFULL

– MVSESA/AMRFCLEAR

JES2 Spool Recovery Processing

The JES2 spool recovery processing function that is provided responds to JES2 spool shortage and spool full messages by JES2 spool recovery processing to downgrade the problem of excessive spool usage.

The JES2 spool recovery processing function that is provided includes the following items:

- Automation routines AOFRSD01, AOFRSD09, AOFRSD0H. See "AOFRSD01" on page 192, "AOFRSD09" on page 195, and "AOFRSD0H" on page 200.
- Automation table entries for system messages HASP050 and HASP355.
- Configuration parameters for the JES2 spool recovery process in the JES2 SPOOLSHORT and JES2 SPOOLFULL policy items of the automation control file.
- Recovery commands defined in the JES2 SPOOLSHORT and JES2 SPOOLFULL policy items of the automation control file.

Drain Processing Prior to JES2 Shutdown

SA z/OS provides functions for drain processing of JES2 resources prior to JES2 shutdown.

The JES2 drain processing function that is provided includes the following items:

- Automation routines AOFRSD07, AOFRSD0F, AOFRSD0G. See "AOFRSD07" on page 194, "AOFRSD0F" on page 197 and "AOFRSD0G" on page 199.
- Automation table entries for system message HASP607.
- Specifications in the JES2 DRAIN automation policy item for the JES2 resources that are to be drained and how they are to be drained prior to JES2 shutdown.

TWS Automation Operation

SA z/OS provides functions to respond to errors with TWS operations and jobs.

The functions that are provided include the following routines and AT entries for associated messages:

- EVJEAC03 and EQQE036I, see "EVJEAC03" on page 225
- EVJEAC04 and EVJ120I, see "EVJEAC04" on page 226
- EVJRAC05 and EQQE026I, see "EVJRAC05" on page 228
- EVJRSJOB and EQQE107I, EQQE107I, and EQQW079W, see "EVJRSJOB" on page 230

IMS Transaction Recovery

SA z/OS provides an IMS transaction recovery function. This responds to an IMS application program abend message by issuing predefined replies or commands from the ACF for recovery purposes. A recovery action is not issued if the program is excluded from recovery processing, or the occurrence of the incoming message exceeds a predefined critical threshold.

The IMS transaction recovery function that is provided by SA z/OS includes the following:

- Automation routine EVIECT0X, see "EVIECT0X" on page 219
- A NetView automation table entry for the application program abend message, DFS554A
- The subsystem that issues the abend message has the following automation policy definitions:
  - Error threshold definitions for minor resource PROG.*progid* or TRAN.*tran*
  - Code definitions in the MESSAGES/USER DATA policy item for the message types ABCODEPROG.*progid*, ABCODEPROG, ABCODETRAN.*tran*, or ABCODETRAN
  - Reply or command specifications in the MESSAGES/USER DATA policy item for the message ID DFS554A

AOFRSA01

Purpose

You can use the AOFRSA01 automation routine to respond to logrec data set nearly full or full messages from your system by issuing commands from the ACF to dump and clear the contents of the logrec data set.

AOFRSA01 keeps track of the incoming logrec data set messages and compares their occurrence with predefined thresholds of infrequent, frequent and critical level. An exceeded threshold is taken as the option to select the appropriate commands according to the first field in the command entry of the entry/type-pair MVSESA/LOGREC in the ACF. If no threshold is exceeded the commands to selection option ALWAYS are issued.

AOFRSA01 should be called from the NetView automation table.

Syntax

Restrictions

- Actions are only taken in AOFRSA01 if the recovery automation flag for LOGREC is on.
- Processing in AOFRSA01 is only done if it is called from NetView automation table by one of the expected messages IFB040I, IFB060E, IFB080E or IFB081I.
- The commands from automation policy to dump and clear the LOGREC data set are only issued if a LOGREC recovery function is not already in progress.

Usage

Automation routine AOFRSA01 is intended to respond to the following messages:

```
IFB040I SYS1.LOGREC AREA IS FULL, hh.mm.ss
IFB060E SYS1.LOGREC NEAR FULL
IFB080E LOGREC DATA SET NEAR FULL, DSN=dsname
IFB081I LOGREC DATA SET IS FULL,hh.mm.ss, DSN=dsn
```

The commands to issue are selected from the command entry of the entry/type-pair MVSESA/LOGREC in the ACF.

If no threshold is reached when one of the expected messages arrive, all commands to entries with no selection option and to selection option ALWAYS are selected. If the threshold at level infrequent is exceeded, all commands to entries with no selection specification option and to selection option INFR are selected. In the same way a level of frequent corresponds to selection option FREQ and a level of critical corresponds to selection option CRIT.

Make sure that the automation routine AOFRSA02 is issued by message IFC001I from the NetView automation table, to indicate the completion of the LOGREC recovery function.

Global Variables

&EHKVAR1

When defining the commands in the ACF to dump and clear the contents of the LOGREC data set, the variable &EHKVAR1 can be used for the name of the LOGREC data set. This variable will be substituted with the complete data set name of the LOGREC data set name.

AOFRSA02

Purpose

You can use the AOFRSA02 automation routine to respond to the initialization message of the LOGREC data set to reset the flag, which indicates that the LOGREC recovery function is in progress

AOFRSA02 should be called from the NetView automation table.

Syntax

Restrictions

- Actions are only taken in AOFRSA02 if the recovery automation flag for LOGREC is on.
- Processing in AOFRSA02 is only done if it is called from NetView automation table.

Usage

Automation routine AOFRSA02 is intended to respond to the following message:

```
IFC001I D=devtyp N=x F=track1* L=track2* S=recd** DIP COMPLETE
```

This is produced during the initialization of the LOGREC data set and describes the limits of the data set.

The flag, indicating that the LOGREC recovery function is in progress, is used by automation routine AOFRSA01.

Examples

This example shows a sample scenario for LOGREC data set processing:

The following entries in the NetView automation table are created automatically to issue the appropriate automation routine when one of the expected messages arrives:

```
IF MSGID = 'IFB040I' | MSGID = 'IFB060E' |
   MSGID = 'IFB080I' | MSGID = 'IFB081I'
THEN
EXEC(CMD('AOFRSA01')ROUTE(ONE %AOFOPRECOPER%));

IF MSGID = 'IFC001I'
THEN
EXEC(CMD('AOFRSA02')ROUTE(ONE %AOFOPRECOPER%));
```

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                              Thresholds Definition
Command ===> ____________________________________________________________________

Entry Type : MVS Component         PolicyDB Name   : DATABASE_NAME
Entry Name : MVS_COMPONENTS        Enterprise Name : YOUR_ENTERPRISE

Resource   : MVSESA.LOGREC

Critical Number  . . . . 3       (1 to 50)
Critical Interval  . . . 00:05   (hh:mm or hhmm, 00:01 to 24:00)

Frequent Number  . . . . 3       (1 to 50)
Frequent Interval  . . . 00:30   (hh:mm or hhmm, 00:01 to 24:00)

Infrequent Number  . . . 3       (1 to 50)
Infrequent Interval  . . 24:00   (hh:mm or hhmm, 00:01 to 24:00)
```

Figure 23. Threshold Definitions for MVS Component LOGREC

```
Pass/Selection Automated Function/'*'
Command Text

MVS S CLRLOG,DSN=&EHKVAR1
```

Figure 24. MESSAGES/USER DATA Policy Item for Entry/Type-Pair MVSESA/LOGREC

Assume that the following message arrives the first time for one day:

```
IFB080E LOGREC DATA SET NEAR FULL, DSN=SYS1.AOC1.MAN3
```

Because none of the defined thresholds is exceeded, the automation routine AOFRSA01 searches for defined commands without selection option and to selection option ALWAYS to be issued. With the control file shown above the command `MVS S CLRLOG,DSN=&EHKVAR1` is selected. Before issuing this command, the variable &EHKVAR1 is substituted by the data set name of the received message resulting in `MVS S CLRLOG,DSN=SYS1.AOC1.MAN3`.

If message IFB080E continues to arrive and the occurrence of the expected messages thus exceeds the infrequent, frequent or critical threshold, the automation routine AOFRSA01 searches for defined commands without selection option and to selection option INFR, FREQ or CRIT to be issued.

Because no command is defined with any selection option, only the defined command with no selection option is selected and issued, as in the previous case.

Message AOF589I, AOF588I or AOF587I is issued in cases, where an infrequent, frequent or critical threshold has been exceeded. These messages indicate that an infrequent, frequent or critical threshold action has been processed.

If the recovery processing for a LOGREC data set is still in progress when an expected error message arrives, the following message is issued:

```
AOF585I 15:45 : RECOVERY OF LOGREC IS ALREADY IN PROGRESS -
```

The recovery process is considered to be finished, when message IFC001I arrives telling that the LOGREC data set has been initialized.

AOFRSA03

Purpose

You can use the AOFRSA03 automation routine to respond to SMF data set full or switch messages from your system. AOFRSA03 issues commands from the ACF to dump and clear the contents of the SMF data set.

AOFRSA03 keeps track of incoming SMF data set messages and compares their occurrence with predefined thresholds at infrequent, frequent and critical levels. An exceeded threshold is taken as the option for selecting the appropriate commands according to the first field in the command entry of the MVSESA/SMFDUMP entry/type pair in the ACF. If no threshold is exceeded the commands that are defined for the selection option ALWAYS are issued.

AOFRSA03 should be called from the NetView automation table.

Syntax

Restrictions

- Actions in AOFRSA03 are only taken if the recovery automation flag for SMFDUMP is on.
- Processing in AOFRSA03 is only done if it is called from the NetView automation table by one of the expected messages: IEE362A, IEE262I, IEE391A or IEE392I.

Usage

Automation routine AOFRSA03 is intended to respond to the following messages:

```
IEE362A SMF ENTER DUMP FOR SYS1.MANn ON ser
IEE362I SMF ENTER DUMP FOR SYS1.MANn ON ser
IEE391A SMF ENTER DUMP FOR DATA SET ON VOLSER ser, DSN=dsname
IEE392I SMF ENTER DUMP FOR DATA SET ON VOLSER ser, DSN=dsname
```

that indicate that the SMF data set is ready to be dumped.

Global Variables

&EHKVAR1

When defining the commands in the ACF to dump and clear the contents of the SMF data set, the variable &EHKVAR1 can be used for the name of the SMF data set. This variable will be substituted with the complete data set name by AOFRSA03 when message IEE391A or IEE392I is received. In case of message IEE362A or IEE362I this variable will be substituted with MANn, the second part of the SMF data set name.

&EHKVAR2

When defining the commands in the ACF to dump and clear the contents of the SMF data set, the variable &EHKVAR2 can be used for the name of the SMF data set. This variable will be substituted with the complete data set name by AOFRSA03 when message IEE391A, IEE392I, IEE362A, or IEE362I is received.

Examples

This example shows a sample scenario for SMF data set processing:

The following entries in the NetView automation table are created automatically to issue the appropriate automation routine when one of the expected messages arrives:

```
IF (MSGID = 'IEE362I' | MSGID = 'IEE362A' |
    MSGID = 'IEE391A' | MSGID = 'IEE392I')
THEN
EXEC(CMD('AOFRSA03')ROUTE(ONE %AOFOPRECOPER%));
```

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                              Thresholds Definition
Command ===> ____________________________________________________________

Entry Type : MVS Component         PolicyDB Name   : DATABASE_NAME
Entry Name : MVS_COMPONENTS        Enterprise Name : YOUR_ENTERPRISE

Resource   : MVSESA.SMFDUMP

Critical Number  . . . . 3       (1 to 50)
Critical Interval  . . . 00:05   (hh:mm or hhmm, 00:01 to 24:00)

Frequent Number  . . . . 3       (1 to 50)
Frequent Interval  . . . 00:30   (hh:mm or hhmm, 00:01 to 24:00)

Infrequent Number  . . . 3       (1 to 50)
Infrequent Interval  . . 24:00   (hh:mm or hhmm, 00:01 to 24:00)
```

Figure 25. Threshold Definitions for MVS Component SMFDUMP

```
Pass/Selection Automated Function/'*'
Command Text

_____ ____________________________________________________________________
MVS S SMFDUMP1,DA=''&EHKVAR1'''
_____ ____________________________________________________________________
```

Figure 26. MESSAGES/USER DATA Policy Item for Entry/Type-Pair MVSESA/SMFDUMP

Assume that the following message arrives the first time on one day:

```
IEE391A SMF ENTER DUMP FOR DATASET ON VOLSER 123, DSN=SYS1.AOC1.MAN3
```

Because none of the defined thresholds has been exceeded, the AOFRSA03 automation routine searches for commands to issue that have been defined without a selection option or with the selection option ALWAYS. With the control file shown above the command `MVS S SMFDUMP1,DA='&EHKVAR1'` is selected. Before issuing this command, the variable &EHKVAR1 is substituted with the data set name from the received message, resulting in `MVS S SMFDUMP1,DA='SYS1.AOC1.MAN3'`.

If message IEE391A continues to arrive and the occurrence of the expected messages thus exceeds the infrequent, frequent or critical thresholds, the AOFRSA03 automation routine searches for commands to issue that have been defined without a selection option or with selection option INFR, FREQ or CRIT.

Because no command has been defined with a selection option, only the command that has been defined without a selection option is selected and issued, as in the previous case.

Message AOF589I, AOF588I or AOF587I is issued in cases where an infrequent, frequent or critical threshold has been exceeded. These messages indicate that an infrequent, frequent or critical threshold action has been processed.

AOFRSA08

Purpose

You can use the AOFRSA08 automation routine to respond to syslog being queued messages by starting an external writer to save the syslog that was queued.

AOFRSA08 keeps track of the incoming syslog queued messages and compares there occurrence with predefined thresholds at level infrequent, frequent and critical. An exceeded threshold is taken as option for selecting the appropriate commands according to the first field in the command entry of the entry/type-pair MVSESA/SYSLOG in the ACF. If no threshold is exceeded the commands to selection option ALWAYS are issued.

AOFRSA08 should be called from the NetView automation table.

Syntax

Restrictions

- Processing in AOFRSA08 is only done if it is called from NetView automation table by the expected message IEE043I.
- Actions are only taken in AOFRSA08 if the recovery automation flag for SYSLOG is on and if the status of JES is UP or HALTED.

Usage

Automation routine AOFRSA08 is intended to respond to the message:

```
IEE043I A SYSTEM LOG DATA SET HAS BEEN QUEUED TO SYSOUT CLASS class
```

which indicates that the system closed the system log (SYSLOG) data set and queued the data set to a SYSOUT class.

The commands to issue are selected from the command entry of the entry/type-pair MVSESA/SYSLOG in the ACF.

If no threshold is reached when one of the expected messages arrive, all commands to entries with no selection option and to selection option ALWAYS are selected. If the threshold at level infrequent is exceeded, all commands to entries with no selection specification option and to selection option INFR are selected. In the same way a level of frequent corresponds to selection option FREQ and a level of critical corresponds to selection option CRIT.

Examples

This example shows a sample scenario for SYSLOG processing:

The following entry in the NetView automation table is created automatically to issue AOFRSA08 as response to the incoming IEE043I message:

```
IF MSGID = 'IEE043I'
THEN
EXEC(CMD('AOFRSA08')ROUTE(ONE %AOFOPRECOPER%));
```

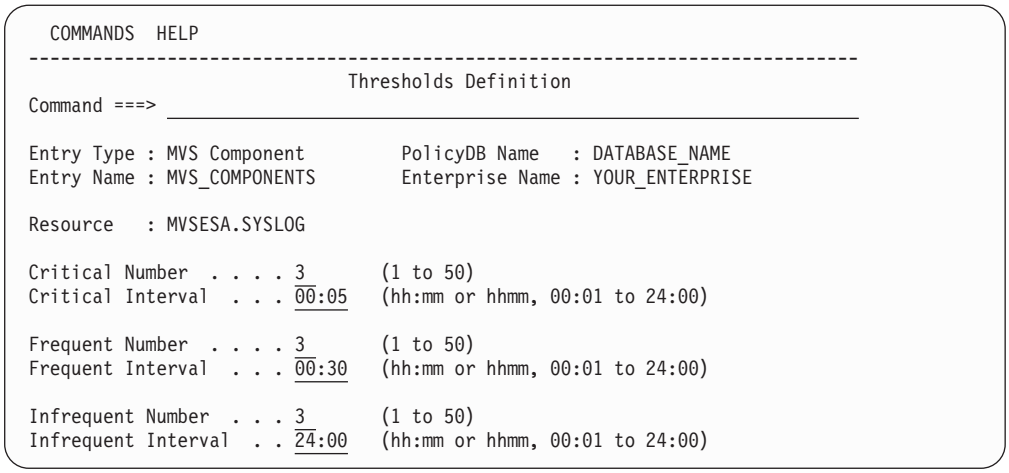


Figure 27. Threshold Definitions for MVS Component SYSLOG

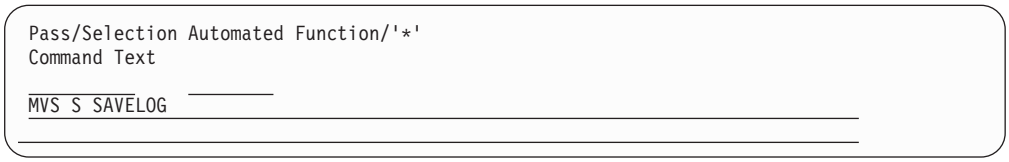


Figure 28. MESSAGES/USER DATA Policy Item for Entry/Type-Pair MVSESA/SYSLOG

Assume that the following message arrives the first time for one day:

```
IEE043I A SYSTEM LOG DATA SET HAS BEEN QUEUED TO SYSOUT CLASS A
```

Because none of the defined thresholds is exceeded, the automation routine AOFRSA08 searches for defined commands without selection option and to selection option ALWAYS to be issued. With the control file shown above the command `MVS S SAVELOG` is selected.

If message IEE043I continues to arrive and the occurrence of the expected messages thus exceeds the infrequent, frequent or critical threshold, the automation routine AOFRSA08 searches for defined commands without selection option and to selection option INFR, FREQ or CRIT to be issued.

Because no command is defined with any selection option, only the defined command with no selection option is selected and issued, as in the previous case.

Message AOF589I, AOF588I or AOF587I is issued in cases, where an infrequent, frequent or critical threshold has been exceeded. These messages indicate that an infrequent, frequent or critical threshold action has been processed.

AOFRSA0C

Purpose

You can use the AOFRSA0C automation routine to respond to a SVC dump taken to a dump data set message by issuing commands from the ACF to format the dump, to clear the dump data sets or to prevent further dumping. The commands to issue are taken from the entry/type-pair MVSESA/MVSDUMP and MVSESA/MVSDUMPTAKEN and selected according to the frequency of the incoming messages and the thresholds defined in the automation policies. The first field in the command entry gives detailed criteria to select the appropriate commands from the ACF.

AOFRSA0C should be called from the NetView automation table.

Syntax

Restrictions

- Actions in AOFRSA0C are only taken if the recovery automation flag for MVSDUMP is on.
- Processing in AOFRSA0C is only done if it is called from NetView automation table by one of the expected messages IEA611I or IEA911E.

Usage

Automation routine AOFRSA0C is intended to respond to the messages:

```
IEA611I {COMPLETE|PARTIAL} DUMP ON dsname
DUMPID=dumpid REQUESTED BY JOB (jobname)
FOR ASIDS(id,id,...)
...

IEA911E {COMPLETE|PARTIAL} DUMP ON SYS1.DUMPnn
DUMPid=dumpid REQUESTED BY JOB (jobname)
FOR ASIDS(id,id,...)
...
```

which indicates that the system wrote a complete or partial SVC dump to an automatically allocated or pre-allocated dump data set on a direct access storage device or a tape volume.

AOFRSA0C keeps track on the reception of these messages and compares the frequency of the incoming messages with predefined thresholds of infrequent, frequent and critical level, where the thresholds to MVS component MVSDUMP are considered. The commands to issue are selected according to the frequency of the incoming messages.

If no threshold is reached, all commands to entries with no selection option and to selection option ALWAYS are selected. If the threshold at level infrequent is exceeded, all commands to entries with no selection option and to selection option INFR are selected. In the same way a level of frequent corresponds to selection option FREQ and a level of critical corresponds to selection option CRIT.

The commands to issue are taken from entry/type-pair MVSESA/MVSDUMP of the ACF with respect to the frequency of the incoming of these messages.

If AOFRSA0C has been triggered on receipt of message IEA911E, additionally all commands from entry/type-pair MVSESA/MVSDUMPTAKEN of the ACF are selected and issued, as long as the critical threshold is not exceeded.

After dump processing has been done, AOFRSA0C further monitors the frequency of messages IEF611I and IEF911E in intervals of 15 minutes. As soon as the frequency falls below the infrequent threshold, all commands of entry/type-pair MVSESA/MVSDUMPRESET are issued.

Global Variables

When defining the commands in the ACF to handle the SVC dump data set, the variables &EHKVAR1 to &EHKVAR6 can be used to be substituted by variable contents of message IEA611I or IEA911E. The variables &EHKVAR1 to &EHKVAR6 are not available in command entries of type MVSDUMPRESET. These variables will be substituted as follows:

&EHKVAR1
dsname of IEA611I or suffix of SYS1.DUMPnn in IEA911E

&EHKVAR2
data set name

&EHKVAR3
dumpid

&EHKVAR4
jobname

&EHKVAR5
ID of address space

&EHKVAR6
dump type (PARTIAL or COMPLETE)

Examples

This example shows the use of automation routine AOFRSA0C in a sample context:

An entry in the NetView automation table is used to issue AOFRSA0C when one of the expected messages arrives:

```
IF MSGID = 'IEA611I' | MSGID = 'IEA911E'
THEN
EXEC(CMD('AOFRSA0C ')ROUTE(ONE %AOFOPRECOPER%));
```

Three threshold levels are defined in the automation policy for MVS component MVSDUMP:

```
AOFKAASR                  SA z/OS  - Command Dialogs
 Domain ID   = IPSNO     ---------- INGTHRES ----------    Date = 08/28/03
 Operator ID = SAUSER                                     Time = 09:38:02


 Specify thresholds and resource changes:

    Resource     =>    MVSESA.MVSDUMP     Group or specific resource
    System       =>    KEY3         System name, domain ID, sysplex name or *all

    Critical     =>    6    errors in  00:30      Time (HH:MM)
    Frequent     =>    4    errors in  00:20      Time (HH:MM)
    Infrequent   =>    2    errors in  00:20      Time (HH:MM)



    Pressing ENTER will set the THRESHOLD values




 Command ===>
    PF1=Help     PF2=End      PF3=Return                       PF6=Roll
                                                             PF12=Retrieve
```

Figure 29. MVSDUMP Thresholds

Automation policy item MESSAGES/USER DATA of entry/type-pair MVSESA/MVSDUMP contains the following command entries with selection options at different levels:

```
Command = ACF ENTRY=MVSESA,TYPE=MVSDUMP,REQ=DISP
SYSTEM = KEY3      AUTOMATION CONFIGURATION DISPLAY - ENTRY= MVSESA
-------------------------------------------------------------------------------
AUTOMATION CONFIGURATION DISPLAY - ENTRY= MVSESA
 TYPE IS MVSDUMP
 CMD             = (FREQ,,'MVS DD ALLOC=INACTIVE')
 CMD             = (INFR,,'MVS DD ALLOC=ACTIVE')
 CMD             = (CRIT,,'MVS DD ALLOC=INACTIVE')
END OF MULTI-LINE MESSAGE GROUP
```

Figure 30. MVSESA/MVSDUMP Command Entries

Automation policy item MESSAGES/USER DATA to entry/type-pair MVSESA/MVSDUMPTAKEN contains the following command entries with no selection options:

```
Command = ACF ENTRY=MVSESA,TYPE=MVSDUMPTAKEN,REQ=DISP
SYSTEM = KEY3      AUTOMATION CONFIGURATION DISPLAY - ENTRY= MVSESA
-------------------------------------------------------------------------------
AUTOMATION CONFIGURATION DISPLAY - ENTRY= MVSESA
 TYPE IS MVSDUMPTAKEN
 CMD             = (,,'MVS DD CLEAR,DSN=&EHKVAR1')
END OF MULTI-LINE MESSAGE GROUP
```

Figure 31. MVSESA/MVSDUMPTAKEN Command Entries

Automation policy item MESSAGES/USER DATA to entry/type-pair MVSESA/MVSDUMPRESET contains the following command entries with no selection options:

```
Command = ACF ENTRY=MVSESA,TYPE=MVSDUMPRESET,REQ=DISP
SYSTEM = KEY3      AUTOMATION CONFIGURATION DISPLAY - ENTRY= MVSESA
-------------------------------------------------------------------------------
AUTOMATION CONFIGURATION DISPLAY - ENTRY= MVSESA
 TYPE IS MVSDUMPRESET
 CMD             = (,,'MVS DD ALLOC=ACTIVE')
END OF MULTI-LINE MESSAGE GROUP
```

Figure 32. MVSESA/MVSDUMPRESET Command Entries

As long as no threshold is exceeded at receipt of one of the IEA611I and IEA911E messages, no action is taken.

If dumps have been taken more often than defined with the infrequent threshold, command `MVS DD ALLOC=ACTIVATE`, specified in entry type MVSDUMP is issued, which makes sure that automatic dump data set allocation is enabled. In case when the dump has been written to a pre-allocated SYS1.DUMP data set, additionally the data set will be cleared by command `MVS DD CLEAR,DSN=&EHKVAR1`, specified in entry type MVSDUMPTAKEN. Variable &EHKVAR1 will be substituted by the numeric suffix of the SYS1.DUMP data set.

The same processing will be done in case, when the incoming dump data set messages exceeds the frequent level.

As soon as the critical threshold is exceeded, the automation routine stops clearing pre-allocated SYS1.DUMP data sets.

After commands having been issued by the automatic processing of dump data sets, automation routine AOFRSA0C checks every 15 minutes whether the infrequent threshold is satisfied again. As soon as this situation is reached, automatic dump data set allocation will be enabled again by command `MVS DD ALLOC=ACTIVE`, as defined in entry type MVSDUMPRESET.

AOFRSA0E

Purpose

Automation routine AOFRSA0E deletes WTORs from SA z/OS display capabilities when they are replied to or canceled.

Syntax

Parameters

id The reply identifiers for cancelled messages.

Restrictions

Processing in AOFRSA0E is only done if it is called from NetView automation table by message IEE400I or IEE600I or if one of these messages are passed by parameter.

Usage

Automation routine AOFRSA0E is intended to respond to the following messages:

```
IEE400I THESE MESSAGES CANCELED- id,id,id
IEE600I REPLY TO id IS; text
```

Message IEE400I says that the system cancelled messages because the issuing task ended or specifically requested that the messages be cancelled. Message IEE600I notifies all consoles that received a message that the system accepted a reply to the message.

As well AOFRSA0E can extract the identifiers of the messages to delete from passed parameters.

Example

The following example shows how to issue AOFRSA0E from the NetView automation table:

```
IF MSGID = 'IEE400I' | MSGID = 'IEE600I'
THEN
EXEC(CMD('AOFRSA0E ')ROUTE(ONE %AOFOPWTORS%));
```

AOFRSA0G

Purpose

You can use the AOFRSA0G automation routine to respond to messages, reporting buffer shortage of the action message retention facility (AMRF) by issuing commands from the ACF to process buffer shortage automation. In case of an incoming buffer shortage message the commands to issue are taken from the entry/type-pair MVSESA/AMRFSHORT with selection option PASS1 and reissued in 1 minute intervals with incremented pass count. In case of buffer full message the commands to issue are taken from entry/type-pair MVSESA/AMRFFULL. If buffer shortage relieved is reported, the commands to entry/type-pair MVSESA/AMRFCLEAR are selected.

AOFRSA0G should be called from the NetView automation table.

Syntax

Restrictions

- Actions are only taken in AOFRSA0G if the recovery automation flag for AMRF is on.
- Processing of system messages in AOFRSA0G is only done if it is called from NetView automation table by message IEA359I, IEA360A or IEA361I.

Usage

Automation routine AOFRSA0G is intended to respond to the messages:

```
IEA359E BUFFER SHORTAGE FOR RETAINED ACTION MESSAGES - 80% FULL
IEA360A SEVERE BUFFER SHORTAGE FOR RETAINED ACTION MESSAGES - 100% FULL
IEA361I BUFFER SHORTAGE RELIEVED FOR RETAINED ACTION MESSAGES
```

IEA359E and IEA360A reports buffer shortage of the buffer area for immediate action messages, non-critical and critical eventual action messages and WTOR messages. IEA361I indicates the reduction of the number of retained action messages so that the buffer is now less than 75% full.

If AOFRSA0G has been triggered on receipt of message IEA359I the commands to issue are taken from entry/type-pair MVSESA/AMRFSHORT, starting at selection option PASS1 and continuing with incremented selection options in 1 minute intervals until message IEA361 reports that buffer shortage has relieved. After arriving the maximal used selection option for a defined command processing restarts at selection option PASS1.

If AOFRSA0G has been triggered on receipt of message IEA360A all commands from entry/type-pair MVSESA/AMRFFULL are issued.

If AOFRSA0G has been triggered on receipt of message IEA361I all commands from entry/type-pair MVSESA/AMRFCLEAR are issued.

Examples

The following example shows a sample scenario for AMRF shortage processing:

Entries in the NetView automation table are used to issue AOFRSA0G when message IEA359E, IEA360E or IEA361I arrives:

```
IF MSGID = 'IEA359I'
THEN
EXEC(CMD('AOFRSA0G')ROUTE(ONE %AOFOPRECOPER%));
IF MSGID = 'IEA360A'
THEN
EXEC(CMD('AOFRSA0G')ROUTE(ONE %AOFOPRECOPER%));
IF MSGID = 'IEA361I'
THEN
EXEC(CMD('AOFRSA0G')ROUTE(ONE %AOFOPRECOPER%));
```

To specify how to respond to message IEA359E and IEA361I, the following command definitions are made in the automation policy under the entry/type-pair MVSESA/AMRFFULL and MVSESA/AMRFCLEAR:

```
Command = ACF ENTRY=MVSESA,TYPE=AMRF*,REQ=DISP
SYSTEM = AOC1      AUTOMATION CONFIGURATION DISPLAY - ENTRY= MVSESA
------------------------------------------------------------------------------
AUTOMATION CONFIGURATION DISPLAY - ENTRY= MVSESA
 TYPE IS AMRFCLEAR
 CMD             = (,,'MVS CONTROL M,AMRF=Y')
 TYPE IS AMRFULL
 CMD             = (,,'MVS CONTROL M,AMRF=N')
END OF MULTI-LINE MESSAGE GROUP
```

Figure 33. MVSESA AMRF Command Definitions

If for example message

```
IEA360A SEVERE BUFFER SHORTAGE FOR RETAINED ACTION MESSAGES - 100% FULL
```

arrives, AOFRSA0G is issued by the shown statement in the NetView automation table, which causes the command `CONTROL M,AMRF=N` to be issued to clear the AMRF buffers.

After AMRF buffer shortage is relieved, the incoming message

```
IEA361I BUFFER SHORTAGE RELIEVED FOR RETAINED ACTION MESSAGES
```

causes command `CONTROL M,AMRF=Y` to be issued to reactivate AMRF.

AOFRSD01

Purpose

You can use the AOFRSD01 automation routine for JES2 spool recovery processing. It responds to JES2 spool shortage messages by initiating the recovery process for JES2 spool shortage. It responds to JES2 spool full messages by initiating the recovery process for JES2 spool full to downgrade the problem of excessive spool usage.

For this purpose AOFRSD01:

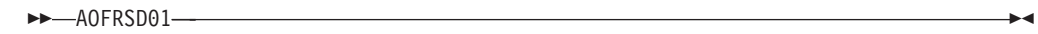
- Makes linear and first order predictions of spool usage, based on actual and historical values
- Posts the spool status to SDF
- Determines the target of recovery process as difference between the actual warning threshold for TG and the buffer value from the ACF. Achieving this target by the recovery process the spool shortage condition will be considered as relieved
- Initiates pass processing to execute the recovery commands of the ACF, defined via policy item JES2 SPOOLSHORT or JES2 SPOOLFULL. The pass processing itself is done by automation routine AOFRSD09 which is issued every retry interval. The retry interval is taken from the ACF.

Recovery commands and configuration parameters like buffer value and retry interval for the JES2 recovery processing can be defined via automation policy item JES2 SPOOLSHORT for spool shortage recovery processing and JES2 SPOOLFULL for spool full recovery processing.

For further information on the automation policy items JES2 SPOOLSHORT and JES2 SPOOLFULL refer to section Defining JES Subsystem in System Automation for z/OS Automation Policy.

AOFRSD01 should be called from the NetView automation table.

Syntax



Restrictions

- Processing in AOFRSD01 is only done if it is called from NetView automation table by JES2 messages HASP050 or HASP355.
- Message HASP355 is only processed if it reports a shortage of track groups (TG).

Usage

Automation routine AOFRSD01 is intended to respond to the following messages:

```
HASP050 JES2 RESOURCE SHORTAGE OF TGs - nnn% UTILIZATION REACHED

HASP355 SPOOL VOLUMES ARE FULL
```

HASP050 indicates that JES2 has a shortage of track groups and the current spool utilization exceeds the current TGWARN value on this JES. TGNWARN is defined in the SPOOLDEF statement in the JES initialization member and can be changed

dynamically. HASP355 indicates that a request for JES2 direct access spool space cannot be processed, because all available space has been allocated to JES2 functions or no spool volumes are available. Therefore the recovery targets in this case are based on a figure of 100% spool utilization.

You should code TGWARN in the SPOOLDEF statement in the JES initialization member so that a SPOOLSHORT recovery will be initiated before a SPOOLFULL condition is reached. If this is not done, the recovery process may become unpredictable. When resetting after a SPOOLFULL condition, the problem is downgraded to a SPOOLSHORT. SA z/OS expects the previously running SPOOLSHORT recovery to activate and try to downgrade the problem to an OK. Without the prior SPOOLSHORT recovery, the spool status will remain in SPOOLSHORT after a successful SPOOLFULL recovery.

The NetView automation table entries for JES2 messages have to respect the one character prefix in front of the message identifier of JES2 messages, identifying the issuing JES.

The spool status is posted to SDF under the SPOOL generic, with the name of the subsystem as its specific name. To get these displayed on an SDF panel, you need status fields for *xxxx*.SPOOL, elements 1 through *n*, where *n* is the number of different subsystems that use the spool.

AOFRSD07

Purpose

You can use the AOFRSD07 automation routine to respond to a JES2 not dormant message during JES2 shutdown by issuing commands for resources that are not drained.

The commands to issue are taken from the automation policy item JES2 DRAIN of application JES2.

Additionally AOFRSD07 calls AOFRSD0F which outputs a list of all active jobs and started tasks and a list of all resources not yet drained.

AOFRSD07 should be called from the NetView automation table.

Syntax

Restrictions

Processing in AOFRSD07 is only done if:

- It is called from NetView automation table by JES2 message HASP607
- The terminate automation flag for JES2 is on
- JES2 is in shutdown progress

AOFRSA07 performs no processing under z/OS 1.7 and above because Console IDs are not valid in that environment.

Usage

Automation routine AOFRSD07 is intended to respond to message

```
HASP607 JES2 NOT DORMANT --  MEMBER DRAINING, RC=rc text
```

which indicates in case the P JES2 command was entered to withdraw JES2 from the system that not all of JES2's functions have completed.

To find out all resources not drained, the response to JES2 command DU,STA is processed. For each resource in status DRAINING the corresponding command from the automation policy item JES2 DRAIN for this resource type to force drain is issued. Resources in status ACTIVE are first stopped with JES2 command P resource, before the command from the automation policy item to force drain is issued. Resources in status INACTIVE are only stopped with JES2 command P resource.

In cases, where the automation is unable to issue actions on not yet drained resources, JES2 is set to status STUCK and a message is issued which tells that an operator action is required. Those situations occur if no command is specified in automation policy item JES2 DRAINED of JES2 to drain a resource or if a not yet drained resource is in an unknown status

AOFRSD09

Purpose

Automation routine AOFRSD09 is used for JES2 spool recovery. It is called by AOFRSD01 via a timer every retry interval to monitor spool utilization of JES2 and to successive issue the recovery commands of policy item JES2 SPOOLSHORT or JES2 SPOOLFULL.

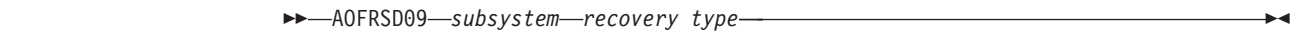
For this purpose AOFRSD09 processes the following steps:

- AOFRSD09 issues the JES2 command D SPOOL to obtain the current spool usage.
- AOFRSD09 re-evaluates the target of recovery process based on the actual warning threshold for TG and the buffer value from the ACF.
- If the recovery target has not yet been achieved and the own JES2 subsystem is responsible for the spool recovery, AOFRSD09 increments the pass count and issues the appropriate commands from the ACF. To determine the responsible JES2 subsystem for spool recovery in a shared JES2 environment, where all JES2 subsystems receive a copy of the spool shortage message, AOFRD09 compares the list of cpuids, defined in ACF, with the response to JES2 command D MEMBER,STATUS=ACTIVE. The first active cpuid of the list is considered to be the responsible JES2 subsystem for spool recovery.
- In case the spool shortage problem has already relieved, AOFRSD09 stops the recovery process and sets a timer to reset the pass count for the recovery commands after the reset interval.

Recovery commands and configuration parameters like buffer value, reset interval and cpuid list for the JES2 recovery processing can be defined via automation policy item JES2 SPOOLSHORT for spool shortage recovery processing and JES2 SPOOLFULL for spool full recovery processing.

For further information on the automation policy items JES2 SPOOLSHORT and JES2 SPOOLFULL refer to section Defining JES Subsystem in System Automation for z/OS Automation Policy.

Syntax



Parameters

subsystem
The subsystem name of JES2. This parameter is required.

recovery type
This parameter is used to distinguish between a JES2 spool shortage and a JES2 spool full condition. This parameter is required.

SHORT
The automatic recovery from a JES2 spool shortage condition is to be processed.

FULL The automatic recovery from a JES2 spool full condition is to be processed.

Restrictions

- Processing of recovery commands in AOFRSD09 is only done if the recovery automation flag for JES2 is on. Otherwise the recovery process is suspended and the pass count for selection recovery commands from the ACF is not incremented.
- Automation routine AOFRSD09 should be processed by JESOPER. If it is called on another task it is routed back to JESOPER.
- Processing in AOFRSD09 is only done if the specified type of spool recovery process has been initiated by automation routine AOFRSD01.
- During a SPOOLFULL recovery condition, the processing for SPOOLSHORT recovery is suspended.

Usage

The recovery commands to issue are selected from the command entry of policy item JES2 SPOOLSHORT or JES2 SPOOLFULL. A pass count is used as selection option and incremented at each successive processing of automation routine AOFRSD09. At initialization of the recovery process, the pass count is set to value PASS1 by automation routine AOFRSD01.

If pass processing runs out of defined recovery commands before the spool shortage condition is resolved, AOFRSD09 re-executes the recovery sequence from PASS1. You can change this behavior by setting the appropriate advanced automation option at start up of System Automation. You can use the AOFSPOOLSHORTCMD variable (for SPOOLSHORT conditions) and the AOFSPOOLFULLCMD variable (for SPOOLFULL conditions) to tell automation routine AOFRSD09 to stop recovery attempts when all commands have been executed and to issue message AOF294I to inform the operator that manual intervention is required in order to resolve the spool condition. For more information on advanced automation options refer to 'Global Variables to Enable Advanced Automation' in System Automation for z/OS: Customization and Programming.

Global Variables

When defining the commands in the SPOOLFULL or SPOOLSHORT processing panel of the ACF to handle the recovery, the variables &EHKVAR1 and &EHKVAR2 can be used to be substituted by variable contents. Variable &EHKVAR1 will be substituted by the current spool utilization and &EHKVAR2 contains the recovery target.

AOFRSD0F

Purpose

Automation routine AOFRSD0F is used by AOFRSD07 for drain processing prior to JES2 shutdown. Every shutdown delay interval, AOFRSD0F displays all JES2 resources not yet drained. For this purpose it scans the response to JES2 command DA,S for executing tasks, the response to JES2 command DA,J for executing jobs and the response to JES2 command DU,STA for started devices or lines not yet drained and displays the result in a message.

Syntax

Parameters

subsystem
The subsystem name of JES2.

Restrictions

Processing in AOFRSD0F is only done

- The subsystem is of type JES2
- JES2 is in shutdown progress
- The terminate automation flag is on

Usage

This automation routine is performed as part of the SHUTDOWN processing.

Examples

This example shows a sample scenario for JES2 drain processing prior to JES2 shutdown.

The following statement shows how AOFRSD07 is issued from the NetView automation table by JES2 message

```
$HASP607:  IF MSGID(2) = 'HASP607'
THEN
EXEC(CMD('AOFRSD07')ROUTE(ONE %AOFOPJESOPER%));
```

Assume the following drain processing specifications in automation policy item JES2 DRAIN:

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                           JES2 DRAIN Specifications
Command ===> ____________________________________________________________

Entry Type : Application           PolicyDB Name   : DATABASE_NAME
Entry Name : JES2                  Enterprise Name : YOUR_ENTERPRISE

Subsystem: JES2
Enter information (Yes or No) for initial drain to bring down JES2 facilities.
  LIN . . . . . YES       Drain lines
  LOG . . . . . YES       Drain JES2-VTAM interface
  OFF . . . . . NO        Drain spool offloaders
  PRT . . . . . YES       Drain printers
  RDR . . . . . YES       Drain readers
  PUN . . . . . YES       Drain punches
Enter information (Command or No) for force drain if normal drain fails.
  LIN . . . . . $E        Force drain lines
  LOG . . . . . $E        Force drain JES2-VTAM interface
  OFF . . . . . NO        Force drain spool offloaders
  PRT . . . . . $I        Force drain printers
  RDR . . . . . $C        Force drain readers
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 34. JES2 DRAIN Specifications Panel

The list of commands to force drain of JES2 resources are passed to entry/type-pair JES2/FORCEDRAIN in the ACF and can be displayed with the DISPACF command:

```
Command = ACF ENTRY=JES2,TYPE=FORCEDRAIN,REQ=DISP
SYSTEM = KEY3      AUTOMATION CONFIGURATION DISPLAY - ENTRY= JES2
------------------------------------------------------------------------------
AUTOMATION CONFIGURATION DISPLAY - ENTRY= JES2
 TYPE IS FORCEDRAIN
 LIN             = ""$E""
 LOG             = ""$E""
 OFF             = ""NO""
 PRT             = ""$I""
 RDR             = ""$C""
 PUN             = ""$E""
END OF MULTI-LINE MESSAGE GROUP
```

Figure 35. DISPACF Panel

Assume that during a shutdown of JES2 message $HASP607 arrives, indicating that not all of JES2's functions have completed and that JES2's response to command `$DU,STATUS` is:

```
$HASP636 13.53.22 $DU,STA
LINE1      UNIT=0FF3,STATUS=ACTIVE/BOEVM9,DISCON=NO
```

Automation routine AOFRSD07 first issues JES2 command `$PLINE1` to stop the line and then issues JES2 command `$E`, according to the policy specifications FOR entry/type-pair JES2/FORCEDRAIN.

Then automation routine AOFRSD0F is executed every shutdown delay interval, to list all JES2 resources not drained.

AOFRSD0G

Purpose

You can use the AOFRSD0G automation routine to drain JES2 resources prior to JES2 shutdown. AOFRSD0G issues commands to drain the initiators, offloader tasks, lines, printers, punches and readers, depending on which resources are listed and enabled in the automation policy item JES2 DRAIN of application JES2.

AOFRSD0G is used by the DRAINJES command.

Syntax

```
►►—AOFRSD0G—subsystem———————————————————————————————————————►◄
```

Parameters

subsystem
The subsystem name of JES2.

Restrictions

- Processing in AOFRSD0G is only done if the subsystem is of type JES2.

Usage

For all resources enabled to initial drain in automation policy item JES2 DRAIN of application JES2 the JES2 command P is issued.

Example

Call `AOFRSD0G JES2` to stop all resources enabled in JES2 DRAIN for init drain.

These resources can be listed with command `DISPACF JES2 INITDRAIN`.

```
Command = ACF ENTRY=JES2,TYPE=INITDRAIN,REQ=DISP
SYSTEM = AOC1      AUTOMATION CONFIGURATION DISPLAY - ENTRY= JES2
------------------------------------------------------------------------------
AUTOMATION CONFIGURATION DISPLAY - ENTRY= JES2
 TYPE IS INITDRAIN
 LIN             = ""YES""
 LOG             = ""YES""
 OFF             = ""NO""
 PRT             = ""YES""
 RDR             = ""YES""
 PUN             = ""YES""
END OF MULTI-LINE MESSAGE GROUP
```

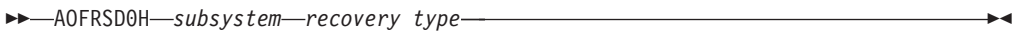
Figure 36. DISPACF JES2 INITDRAIN Panel

AOFRSD0H

Purpose

Automation routine AOFRSD0H is used for JES2 spool recovery. It is called by AOFRSD09 via a timer command after the reset interval and cleans up the pass counter for the pass processing of the recovery commands of the ACF.

Syntax



Parameters

subsystem
The subsystem name of JES2. This parameter is required.

recovery type
This parameter is used to distinguish between a JES2 spool shortage and a JES2 spool full condition. This parameter is required.

SHORT
The pass counter for spool shortage recovery processing is to be reset.

FULL The pass counter for spool full recovery processing is to be reset.

Restrictions

- Automation routine AOFRSD0H should be processed by JESOPER. If it is called on another task it is routed back to JESOPER.
- Each recovery action during the reset interval
- AOFRSD0H is only scheduled after the reset interval if no new recovery action of the corresponding type SHORT or FULL has been taken during this time.
- The pass counter for spool full recovery processing is reset by AOFRSD0H after the reset interval, even if spool short recovery is still in progress.

Examples

The following example shows a sample scenario for JES2 spool recovery processing:

The following entries in the NetView automation table are used to issue automation routine AOFRSD01 from the NetView automation table, when one of the expected messages arrives:

```
IF MSGID(2) = 'HASP050' & TEXT = .'TGS'.
THEN
EXEC(CMD('AOFRSD01')ROUTE(ONE %AOFOPJESOPER%));
IF MSGID(2) = 'HASP355'
THEN
EXEC(CMD('AOFRSD01')ROUTE(ONE %AOFOPJESOPER%));
```

The SPOOLSHORT recovery is configured via automation policy item JES2 SPOOLSHORT as shown in Figure 37 on page 201.

```
   COMMANDS  HELP
------------------------------------------------------------------------------
                              SPOOLSHORT Processing
Command ===> ____________________________________________________________________

Entry Type : Application           PolicyDB Name   : DATABASE_NAME
Entry Name : JES2                  Enterprise Name : YOUR_ENTERPRISE

Enter SPOOLSHORT settings.

Retry Time . . . . 00:05:00      Spool recovery attempt interval (hh:mm:ss)
Buffer . . . . . . 5             Recovery target below TGWARN (0->50)
Reset Time . . . . 00:15:00      Recovery reset interval (hh:mm:ss)

Priority of systems for spool recovery:

CPUID  1 ____   2 ____   3 ____   4 ____   5 ____   6 ____   7 ____   8 ____
       9 ____  10 ____  11 ____  12 ____  13 ____  14 ____  15 ____  16 ____
      17 ____  18 ____  19 ____  20 ____  21 ____  22 ____  23 ____  24 ____
      25 ____  26 ____  27 ____  28 ____  29 ____  30 ____  31 ____  32 ____

Edit Spoolshort Pass Commands . . YES    YES  NO
```

Figure 37. JES2 SPOOLSHORT Recovery Definition

Because no cpuids are defined, the own JES2 subsystem is responsible for JES2 spool recovery processing. Entering YES in field Edit Spoolshort Pass Commands allows you to edit the pass recovery commands that are defined as shown by the following response panel to command DISPACF JES2:

```
Command = ACF ENTRY=JES2,TYPE=*,REQ=DISP
SYSTEM = KEY3      AUTOMATION CONFIGURATION DISPLAY - ENTRY= JES2
------------------------------------------------------------------------------
 TYPE IS SPOOLSHORT
 CMD             = (PASS1,,'MVS $PQ,Q=N,A=3')
 CMD             = (PASS1,,'MVS $OQ,Q=N,A=3,CANCEL')
 CMD             = (PASS1,,'MVS $PQ,Q=V,A=3')
 CMD             = (PASS1,,'MVS $OQ,Q=V,A=3,CANCEL')
 CMD             = (PASS2,,'MVS $PQ,ALL,A=4')
 CMD             = (PASS2,,'MVS $OQ,ALL,A=4,CANCEL')
 CMD             = (PASS3,,'MVS $PQ,ALL,A=3')
 CMD             = (PASS3,,'MVS $OQ,ALL,A=3,CANCEL')
 CMD             = (PASS4,,'AORRSPLS RANGE=JOB1-5000,NAME=T*')
 CMD             = (PASS4,,'AORRSPLS RANGE=JOB5000-10000,NAME=T*')
 CMD             = (PASS4,,'AORRSPLS RANGE=JOB10000-15000,NAME=T*')
 CMD             = (PASS4,,'AORRSPLS RANGE=JOB15000-20000,NAME=T*')
```

Figure 38. DISPACF Command Response Panel

Assume, a JES2 spool shortage problem is reported by message
`$HASP050 JES RESOURCE SHORTAGE OF TGS - 80% UTILIZATION REACHED`

issuing automation routine AOFRSD01 by the appropriate NetView automation table entry, which initiates the JES2 SPOOLSHORT recovery process and sets an every timer, to call the pass processing routine by issuing AOFRSD09 JES2 SHORT every 5 minutes, as defined in the customization dialog for SPOOLSHORT processing shown above.

AOFRSD09 redetermines the actual spool usage, compares it with the defined TGWARN of 80% and calculates the target of recovery as difference of TGWARN and the buffer value resulting in a value of 75.

If this value is exceeded by the actual spool usage, all recovery commands with selection option PASS1 of the ACF to recovery type SPOOLSHORT are issued. After the retry interval of 5 minutes, AOFRSD09 is re-issued again by the timer.

If AOFRSD09 now determines that the JES2 spool shortage problem has been relieved, it stops recovery processing and sets a timer to issue `AOFRSD0H JES2 SHORT` after the reset interval of 15 minutes.

If none of the expected JES2 messages arrives by the end of the reset interval, automation routine AOFRSD0H resets the pass count to 1, so that the next SPOOLSHORT recovery process issues recovery commands beginning again at selection option PASS1.

EVEEARMW

Purpose

This routine will reply to the DFHKE0408D message. This message is issued when CICS is started with TYPE=COLD or INITIAL and attempts to register with ARM and fails to do so. The appropriate reply is ASIS or AUTO.

This routine should be invoked from the NetView automation table.

Syntax

Usage

The automation routine EVEEARMW is intended to respond to the following message.

```
DFHKE0408D applid PLEASE SPECIFY START TYPE, 'ASIS' OR 'AUTO'.
```

The automation routine will reply with the data specified in the MESSAGES/USER DATA reply policy for the message. The Selection field is set to ARMSTART if CICS was started by ARM or NOARMSTART if CICS was started manually.

EVEED004

Purpose

This routine will post or remove Link Monitoring events for links that are defined to be monitored to SDF or NMC, or both. The resource name formats are defined in *IBM Tivoli System Automation for z/OS CICS Automation Programmer's Reference and Operator's Guide*.

This routine should be invoked from the NetView automation table.

Syntax

Parameters

message
The full message text from the NetView automation table trap.

Usage

The automation routine EVEED004 is intended to respond to the following messages for Link recovery:

```
EVE811I procname : connid - Critical connection to applid in trouble.
EVE812I procname : connid - Connection to applid in trouble.
EVE813I procname : connid - Critical conn. to applid repaired, actions = count1
EVE814I procname : connid - Connection to applid repaired, actions = count1.
EVE815I procname : connid - Critical connection to applid is up.
EVE816I procname : connid - Connection to applid is up.
EVE817I procname : connid - Critical connection to applid, AV=count1 MX=count2 AC=count3.
EVE818I procname : connid - Connection to applid, AV=count1 MX=count2 AC=count3.
EVE819I procname : connid - Critical connection to applid is down.
EVE820I procname : connid - Connection to applid is down.
```

EVEEI004

Purpose

This routine will determine the version number of the CICS subsystem being started.

This routine should be invoked from the NetView automation table.

Syntax

Parameters

version CICS version in *version.release.modlevel* format.

Usage

The automation routine EVEEI004 is intended to process the following message.

```
DFHSI1500 applid element startup is in progress for CICS Transaction Server Version version
```

The automation routine will get the version number and store it for later use.

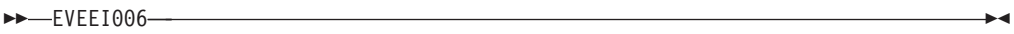
EVEEI006

Purpose

This routine will determine the start type of the CICS subsystem being started.

This routine should be invoked from the NetView automation table.

Syntax



Usage

The automation routine EVEEI006 is intended to process the following message.

```
DFHSI1502I applid CICS startup is {Cold | Warm | Emergency | Initial}.
```

EVEEI009

Purpose

This routine will set the CICS, CICSPlex CMAS or the PPI associated with the CICS subsystem to UP.

This routine should be invoked from the NetView automation table.

Syntax

Parameters

PPI Specifies that the PPI not the CICS Subsystem is in UP state.

Usage

The automation routine EVEEI009 is intended to process the following messages.

```
DFHSI1517 applid Control is being given to CICS.
EYUXL0010I CMAS initialization complete
EVE172I applid : PPI active.
```

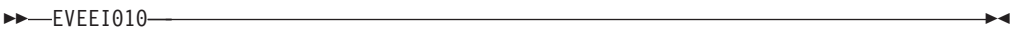
EVEEI010

Purpose

This routine will start active monitoring for Link Monitoring and Health Checking.

This routine should be invoked from the NetView automation table.

Syntax



Usage

The automation routine EVEEI010 is intended to process the following message.

```
EVE172I applid : PPI active.
```

EVEEI115

Purpose

This routine will reply to the DFHPA1104 message. This message is issued to get system start up parameters.

This routine should be invoked from the NetView automation table.

Syntax

Usage

The automation routine EVEEI115 is intended to respond to the following messages.

```
DFHPA1104 applid SPECIFY ALTERNATIVE SIT PARAMETERS, IF ANY, AND THEN TYPE '.END'.
DFHPA1105 applid CONTINUE SPECIFYING SIT PARAMETERS AND THEN TYPE '.END'.
```

The automation routine will reply with the data specified in the INGREQ command, INGSET command or the CICSOVRD command. The START= parameter is always specified and is the start type as coded in the INGREQ command. Data from the INGREQ applparms field is appended. Alternatively, data from the CICSOVRD command or INGSET command is appended. Please note, there is only one field for this override, order of processing is INGREQ applparms take precedence over INGSET or CICSOVRD which both update the same field.

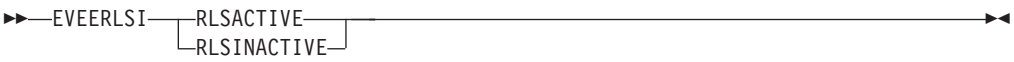
EVEERLSI

Purpose

This routine is a routine to track the CICS VSAM RLS status.

This routine should be invoked from the NetView automation table.

Syntax



Parameters

RLSACTIVE
Specifies that CICS is using RLS.

RLSINACTIVE
Specifies that CICS is not using RLS.

Usage

The EVEECO05 automation routine is intended to respond to the following messages for VSAM RLS status processing:

```
DFHFC0153 applid The previous instance of the SMSVSAM server has failed. File
          control RLS access is being closed down.
DFHFC0501 applid RLS OPEN of file filename failed. VSAM has returned code 16 in
          R15. RLS access has been disabled.
DFHFC0504 applid RLS OPEN of file filename failed. The VSAM SHOWCB macro has
          detected a RLS VSAM server failure. RLS access has been disabled.
DFHFC0508 applid RLS OPEN of file filename failed. VSAM has returned code X'AA'
          in register 15. RLS access has been disabled.
DFHFC0563 applid The RLS control ACB has been successfully unregistered by CICS.
DFHFC0562 applid The RLS control ACB has been successfully registered by CICS.
DFHFC0564 applid The register of the RLS control ACB has failed. VSAM macro
          IDAREGP return code X'rrrr', reason code X'cccc', error data X'dddd'.
DFHFC0565 applid The unregister of the RLS control ACB has failed. VSAM macro
          IDAUNRP return code X'rrrr', reason code X'cccc', error data X'dddd'.
DFHFC0566 applid The register of the RLS control ACB has failed. VSAM macro
          IDAREGP return code X'rrrr', reason code X'cccc'.
DFHFC0567 applid The unregister of the RLS control ACB has failed. VSAM macro
          IDAUNRP return code X'rrrr', reason code X'cccc'.
DFHFC0571 applid RLS access cannot be restarted.
DFHFC0570 applid File control RLS access has been enabled.
DFHFC0577 applid RLS offsite recovery is now complete. RLS access is allowed.
```

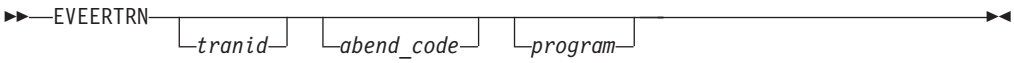
EVEERTRN

Purpose

This routine will handle transaction recovery.

This routine should be invoked from the NetView automation table.

Syntax



Parameters

tranid Optional transaction ID that has abended.

abend_code
Optional abend code.

program
Optional program name of abending program.

If no parameters are specified, these values are derived from the message that invoked this routine.

Usage

The automation routine EVEERTRN is intended to respond to the following messages.

```
DFHAC2231 date time applid Transaction tranid running program program name term
          termid has lost contact with its coordinator system during syncpoint and
          has abended with code ASP1. The unit of work is shunted until contact is
          restored{. EXCI job = }exci_id. condmsg
DFHAC2232 date time applid Transaction tranid running program program name term
          termid has lost contact with its coordinator system during syncpoint and
          has abended with code ASPO. All updates will be unilaterally
          committed{. EXCI job = }exci_id. condmsg
DFHAC2233 date time applid Transaction tranid running program program name term
          termid has lost contact with its coordinator system during syncpoint and
          has abended with code ASPP. All updates will be unilaterally backed
          out{. EXCI job = }exci_id. condmsg
DFHAC2236 date time applid Transaction tranid abend secondary abcode in program
          program name term termid. Updates to local recoverable resources will be
          backed out{. EXCI job = }exci_id. condmsg
DFHAC2245 date time applid A CICS-generated syncpoint request could not be completed
          normally because a connected system has requested that the unit of work be
          rolled back. Transaction tranid running program program name term termid
          has been abnormally terminated with code ASPF{. EXCI job = }exci_id. condmsg
DFHAC2246 date time applid Transaction termination processing could not be completed
          normally because a connected system has requested that the unit of work be
          rolled back. Transaction tranid, terminal termid has been abnormally
          terminated with code ASPN{. EXCI job = }exci_id. condmsg
DFHAC2247 date time applid Transaction tranid running program program name term
          termid has requested rollback, but was using a type of processing for which
          rollback is not supported. The transaction has been abnormally terminated
          with code ASP8 {. EXCI job = }exci_id. condmsg
DFHAC2248 date time applid Transaction tranid running program program name term
          termid has failed with abend ASP7 following the failure of a local
          resource owner in the prepare phase of syncpoint. Updates will be backed
          out{. EXCI job = }exci_id. condmsg
DFHAC2249 date time applid Transaction tranid running program program name term
```

```
          termid has failed with abend ASP7 following the failure of a remote system
          in the prepare phase of syncpoint. Updates will be backed
          out{. EXCI job = }exci_id. condmsg
DFHAC2251 date time applid Transaction tranid running program program name term
          termid has failed with abend ASPQ. Syncpoint commit processing has failed
          while communicating with a remote system{. EXCI job = }exci_id. condmsg
DFHAC2252 date time applid Transaction tranid in program program name term termid
          has lost contact with its coordinator system during syncpoint processing.
          No updates have been performed by this system; it has abended with code
          ASPR{. EXCI job = }exci_id. condmsg
DFHAC2253 date time applid Transaction tranid running program program name term
          termid has failed with abend ASP2 due to the links to the remote systems
          being in an invalid state. Updates will be backed out{. EXCI job = }exci_id.
          condmsg
```

Note, these messages are not normally issued to the system console, so if transaction recovery is required the messages should be included in the MESSAGES/USER DATA policy so that the CICS message exit will force CICS to WTO them.

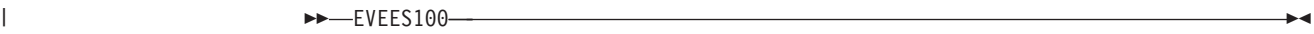
EVEES100

Purpose

This generic routine performs CICS short on storage recovery.

This routine should be invoked from the NetView automation table.

Syntax



Usage

The EVEES100 automation routine is intended to respond to the following messages for CICS Short on Storage events:

```
DFHSM0131 applid CICS is under stress (short on storage below 16MB).
DFHSM0132 applid CICS is no longer short on storage below 16MB.
DFHSM0133 applid CICS is under stress (short on storage above 16MB).
DFHSM0134 applid CICS is no longer short on storage above 16MB.
```

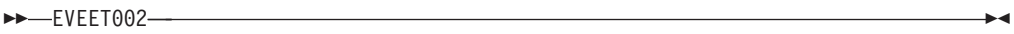
EVEET002

Purpose

This routine will process messages to handle the status of the UOW processing.

This routine should be invoked from the NetView automation table.

Syntax



Usage

The automation routine EVEET002 is intended to respond to the following message.

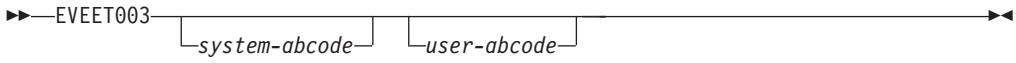
```
DFHRM0130 applid Recovery manager has successfully quiesced.
```

EVEET003

Purpose

This routine is a part of the Abend Recovery for CICS Regions. It is intended to be invoked from the NetView automation table.

Syntax



Parameters

system-abcode
Optional System abend code.

user-abcode
Optional User abend code.

Usage

The automation routine EVEET003 is intended to respond to messages:

```
DFHCC0001 applid An abend (code aaa/bbbb) has occurred at offset X"offset" in
          the {local | global} catalog, module modname
DFHPC0401 applid Abend abcode issued by yyy task.
DFHPC0405 applid Abend abcode2 has been issued while processing abend abcode1
          for the same task, transaction tranid.
DFHPC0408 applid Abend abcode has been issued during post commit processing,
          transaction tranid.
DFHPC0409 applid Abends abcode2 and abcode3 have been issued while processing
          abend abcode1 for the same task, transaction tranid.
DFHSR0601 applid Program interrupt occurred with system task taskid in control
DFHSR0602 applid Program interrupt routine has been entered while processing
          program interrupt for same task
DFHSR0603 applid Program interrupt has occurred
DFHSR0605 applid Error from KE Domain - DFHSRP initialization
DFHSR0606 applid Abend (code aaa/bbbb) has been detected.
DFHSR0612 applid Abend recovery has been entered by same task
DFHSR0613 applid Abend has occurred with system task taskid in control
DFHSR0615 applid Program interrupt has occurred in recovery task
DFHTC1001 applid Terminal control initialization failed (modname).
DFHTM1797 applid System termination program has abended.
DFHDM0106 applid The Domain Manager records on the CICS Catalog may have been
          corrupted.
DFHKE1800 applid ABNORMAL TERMINATION OF CICS IS COMPLETE.
DFHLG0736 applid A failure has occurred while reading from the system log
          (journalname). The requested data could not be found. CICS will be
          quiesced allowing some tasks to complete. Further work requires an
          initial start.
DFHLG0738 applid A failure has occurred while reading the system log (journalname).
          The requested data could not be found. CICS will be terminated. Further
          work requires an initial start.
DFHLG0740 applid While writing data to the system log (journalname), a lost data
          warning was received. CICS will be quiesced without logging, allowing
          tasks to complete. Further work requires an initial start.
DFHSI1542 applid Takeover by the CICS alternate system has failed. Emergency
          restart could not be performed.
```

This informs an operator that a CICS region has abended.

This routine will stop health check monitors and write an SMF record (optionally) to log the abend.

In addition the routine does a CDEMATCH for message ID ABCODSYSTM with *code1* set to the *messageid* that invoked the routine, *code2* set to the *system-abcode* and *code3* set to the user-abcode. The result of the match is used as the determine the parameters to TERMMSG. If NORESTART is returned then the BREAK=YES parameter is used on TERMMSG. If RESTART is returned then the ABEND=YES parameter is used on TERMMSG. If neither of these values are returned, NO TERMMSG command is invoked.

EVEETUOW

Purpose

This routine will process messages to handle the status of the UOW processing. It provides two services:

1. Detecting if units of work are outstanding at CICS shutdown and optionally forcing an AUTO start for the next start up of the CICS subsystem.
2. Detecting if an INITIAL start is required and optionally forcing an INITIAL start at the next start up of the CICS subsystem.

This routine should be invoked from the NetView automation table.

Syntax

Parameters

INITIAL
Optional, specified when INITIAL restart is to be forced.

Usage

The automation routine EVEETUOW is intended to respond to the following messages.

```
DFHDM0106 applid The Domain Manager records on the CICS Catalog may have been
          corrupted.
DFHLG0736 applid A failure has occurred while reading from the system log
          (journalname). The requested data could not be found. CICS will be
          quiesced allowing some tasks to complete. Further work requires an
          initial start.
DFHLG0738 applid A failure has occurred while reading the system log (journalname).
          The requested data could not be found. CICS will be terminated. Further
          work requires an initial start.
DFHLG0740 applid While writing data to the system log (journalname), a lost data
          warning was received. CICS will be quiesced without logging, allowing
          tasks to complete. Further work requires an initial start.
DFHRM0130 applid Recovery manager has successfully quiesced.
DFHRM0134 applid Recovery manager domain failed reading the global catalog, or
          did not find its control record.
DFHRM0136 applid The applid has changed from old_applid to new_applid. Recovery
          cannot continue.
DFHRM0144 applid Recovery manager catalog record indicates that no recovery is
          possible. An initial start is required.
DFHRM0203 applid There are indoubt_uows indoubt, cfail_uows commit-failed and
          bfail_uows backout-failed UOWs.
DFHRM0204 applid There are no indoubt, commit-failed or backout-failed UOWs.
DFHRM0400 applid A unit of work was incompletely reconstructed from the system log.
DFHRM0401 applid There is no system log or an empty system log has been detected.
```

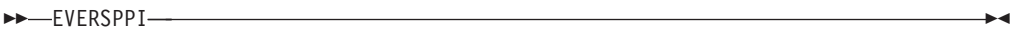
EVERSPPI

Purpose

This routine sets the shutdown status for the PPI running in a CICS address space.

This routine should be invoked from the NetView automation table.

Syntax



Usage

The automation routine EVERSPPI is intended to respond to the following messages for PPI shutdown processing:

```
EVE173I applid : PPI inactive.
```

EVIECT0X

Purpose

This routine can be used to perform recovery processing for transaction and program abends in response to the application program abend message DFS554A.

The routine performs the following actions:

- Parses all data from the DFS554A message
- Checks the appropriate automation flag
- Checks the code definitions for exclusions from recovery for the message types ABCODETRAN or ABCODEPROG
- Performs threshold checking for the triggering DFS554A message

The minor resource name that is used for automation flag checking and threshold checking is either TRAN.*tran* or PROG.*progid*.

If allowed, the recovery commands or replies for message DFS554A with the selection names PROG or TRAN are issued.

EVIECT0X should be called from the NetView automation table.

Syntax

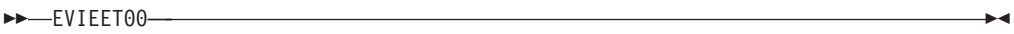
EVIEET00

Purpose

This routine is a generic routine to process IMS TCO Automation.

This routine should be invoked from the NetView automation table.

Syntax



Usage

The automation routine EVIEET00 is intended to respond to the following messages:

```
DFS3343E CANNOT PROCESS DFSTCF LOAD COMMAND, REASON=xx
DFS3350E TCO ABNORMALLY TERMINATED, SEE DUMP
DFS3351E TCO ABNORMALLY TERMINATED, SYSTEM ABEND, SEE DUMP
DFS3613I xxx TCB INITIALIZATION COMPLETE.
```

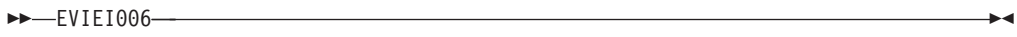
EVIEI006

Purpose

This routine handles the IMS control region restart errors.

This routine should be invoked from the NetView automation table.

Syntax



Usage

The automation routine EVIEI006 is intended to respond to the following messages:

```
DFS166 CHECKPOINT ID NOT ON LOG RE-ENTER RESTART COMMAND
DFS033I DUPLICATE ENTRY ON SIGNON REQUEST, RESTART ABORTED
DFS0618A A RESTART OF A NON-ABNORMALLY TERMINATED SYSTEM MUST SPECIFY EMERGENCY
         BACKUP OR OVERRIDE.
DFS3131I A COLD START OR EMERGENCY RESTART REQUIRED
DFS3626I RESTART HAS BEEN ABORTED
```

EVIEI00Q

Purpose

This routine handles the IMS control region start up initializing. It resets IMS information held by System Automation and gathers the version number of the IMS control region.

This routine should be invoked from the NetView automation table.

Syntax

Usage

The automation routine EVIEI00Q is intended to respond to the following messages:

```
DFS3410I DATA SETS USED ARE DDNAME 'acblib-name' 'format-name' 'MODBLKS-name'
         (time/date stamps if they exist)
INGI1010I Automated Operator Exit Initialized for IMS Level vrm .
```

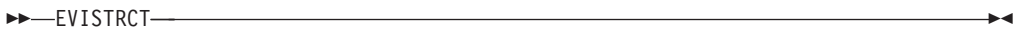
EVISTRCT

Purpose

This routine will Post the IMS sysplex event to both SDF and NMC.

This routine should be invoked from the NetView automation table.

Syntax



Usage

The automation routine EVISTRCT is intended to respond to the following message:

CQS0205E STRUCTURE *structurename* IS FULL

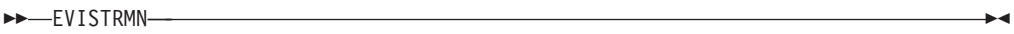
EVISTRMN

Purpose

This routine will reset the posted IMS sysplex event to both SDF and NMC.

This routine should be invoked from the NetView automation table.

Syntax



Usage

The automation routine EVISTRMN is intended to respond to the following message:

CQS0206I CQS *structurename percentage* BELOW THRESHOLD LEVEL

EVJEAC03

Purpose

This routine is called when message EQQE036I is trapped. This message is issued by TWS when a TWS operation has detected a job error. This causes an entry to be added to SDF and the error situation to be posted to NMC.

The EVJEAC03 routine should be called from the NetView automation table.

Syntax

Usage

The automation routine EVJEAC03 is intended to respond to message:

```
EQQE036I JOB JOBNAME(JNUM), OPERATION (OPERNUM) ENDED IN ERROR EC.
         PRTY=PRI, APPL = APPL, WORK STATION = WSID, IA = IA
```

This requests the operator to perform error recovery actions for the current job.

EVJEAC04

Purpose

This routine is called when message EVJ120I is trapped. The message is issued by SA z/OS when a TWS operation has been put into or reset from TWS error status.

The EVJEAC04 routine should be called from the NetView automation table.

Syntax

Usage

The automation routine EVJEAC04 is intended to respond to message:

```
EVJ120I applid iatime opnum job status wsname errcode
        abcode usrcode
```

This causes a Status Display Facility update and an NMC update to occur.

For an operation changing to error status the update will add an entry to SDF and NMC while an operation changing from error status will remove an entry from SDF and NMC.

SDF entries are added to the OPC Automation Application in Error panel (OPCERR).

EVJEOBSV

Purpose

This routine is used to start and stop the TWS status observer.

The EVJEOBSV routine called from within the Policy definitions when starting or stopping the status observer. It is also called internally at SA z/OS initialization time and when an automation manager takeover has been completed as indicated by message HSAM1309I.

Syntax

Parameters

START
Establishes the subscription for the list of special resources defined in the policy.

STOP Removes the subscription.

EVJRAC05

Purpose

This routine is called when message EQQE026I is trapped. This message is issued by TWS when a TWS operation has detected a job error. This causes an entry to be added to SDF and the error situation to be posted to NMC.

The EVJRAC05 routine should be called from the NetView automation table.

Syntax

Usage

The automation routine EVJRAC05 is intended to respond to message:

```
EQQE026I APPLICATION APPL ENDED IN ERROR EC. OPER = OPERNUM,
         PRTY = PRI, IA = IA
```

This requests the operator to perform error recovery actions for the current job.

EVJRSACT

Purpose

This routine keeps track of whether or not the TWS controller is active or in standby. The information is stored in the automation manager.

The routine is called when trapping the following messages:

- EQQN013I
- EQQZ128I
- EQQZ201I

Syntax

EVJRSJOB

Purpose

This routine is called when trapping the following messages:

- EQQE107I
- EQQW079W
- EQQE037I

These messages are issued by TWS when the state of a batch job has changed. This causes an entry to be added to SDF and the error situation to be posted to NMC.

The EVJRSJOB routine should be called from the NetView automation table.

Syntax

Usage

The automation routine EVJRSJOB is intended to respond to messages:

```
EQQE107I OPC-WLM SUCCESSFULLY PROMOTED JBNAM: JBNUM IN
         HI PERFORMANCE CLASS

EQQW079W JBNAM WILL NOT BE SUBMITTED TO WLM FOR
         PROMOTION. WLM REQUEST IS TOO OLD

EQQE037I  JOB JOBNAME(JNUM), OPERATION (OPERNUM) IN
         APPLICATION APPL, IS LATE, WORK STATION = WSID,
         IA = ARRTIME
```

This requests the operator to investigate what is keeping the job from starting and take appropriate actions to enable it to start.

HASP099

Restrictions

Shutdown processing of the JES2 message HASP099 is only done if:

- Shutdown automation for JES2 is on
- JES2 is in the process of being shut down

Usage

The generic routine ISSUEACT responds to message:

```
HASP099 ALL AVAILABLE FUNCTIONS COMPLETE
```

This indicates that all JES2 job processors have become dormant, and no JES2 RJE lines are active.

INGRX740

Purpose

You can use the INGRX740 automation routine to respond to some syslog related system messages by issuing defined recovery actions from the automation control file to restart the syslog or to assign the syslog as a hardcopy medium.

INGRX740 keeps track of the incoming IEE037D syslog inactive message and compares its occurrence with predefined thresholds for the MVS component minor resource, LOG. As long as the critical threshold level is not exceeded, a recovery action related to a previously received system message is issued.

If one of the messages IEE043I, IEE533E or IEE769E is received prior to the IEE037D message that is currently being processed, the commands that have been defined for IEE043I, IEE533E or IEE769E in the entry/type-pair MVSESA/*msgid* of the ACF are issued. If none of these messages has been received prior to the IEE037D message that is currently being processed, the command MVS WRITELOG START is issued.

The recovery routine INGRX740 also responds to an incoming IEE041I message if this indicates that the SYSLOG data set is available for use as a hardcopy log. Commands are issued in response to message IEE041I that are defined in the entry/type-pair MVSESA/IEE041I of the ACF. An appropriate command in this case would be MVS VARY SYSLOG,HARDCPY to have the SYSLOG receive the hardcopy log.

INGRX740 should be called from the NetView automation table.

Syntax

Restrictions and Limitations

Processing in routine INGRX740 is only done if the following conditions are met:

- The recovery automation flag for LOG is on.
- The routine is running on an automation task.
- The routine is called from NetView automation table by one of the expected messages
  - IEE037D
  - IEE041I
  - IEE533E
  - IEE769E
  - IEE043I

Actions in response to message IEE037D are only taken in INGRX740, if the Job Entry Subsystem is up and running.

Usage

Automation routine INGRX740 responds to the following messages:

```
IEE037D LOG NOT ACTIVE
IEE041I THE SYSTEM LOG IS NOW ACTIVE[-MAY BE VARIED AS HARDCOPY LOG]
IEE043I A SYSTEM LOG DATA SET HAS BEEN QUEUED TO SYSOUT CLASS class
IEE533E SYSTEM LOG INITIALIZATION HAS FAILED
IEE769E SYSTEM ERROR IN SYSTEM LOG
```

Example

This example shows a sample scenario for system log failure recovery.

The following entry in the NetView automation table is provided by SA z/OS to issue INGRX740 in response to incoming messages IEE043I and IEE037D:

```
IF MSGID = 'IEE037D' THEN
EXEC(CMD('INGRX740')ROUTE(ONE %AOFOPRECOPER%));
IF MSGID = 'IEE043I' THEN
EXEC(CMD('INGRX740')ROUTE(ONE %AOFOPRECOPER%));
```

Assume that the following threshold levels are defined in the automation policy for MVS component minor resource, LOG.

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                              Thresholds Definition
Command ===> ____________________________________________________________

Entry Type : MVS Component         PolicyDB Name   : DATABASE_NAME
Entry Name : MVS_COMPONENTS        Enterprise Name : YOUR_ENTERPRISE

Resource   : MVSESA.LOG

Critical Number  . . . . 3       (1 to 50)
Critical Interval  . . . 00:05   (hh:mm or hhmm, 00:01 to 24:00)

Frequent Number  . . . . 3       (1 to 50)
Frequent Interval  . . . 00:30   (hh:mm or hhmm, 00:01 to 24:00)

Infrequent Number  . . . 3       (1 to 50)
Infrequent Interval  . . 24:00   (hh:mm or hhmm, 00:01 to 24:00)
```

Figure 39. Threshold Definitions for MVS Component LOG

Assume that a command is defined for message IEE043I in the automation policy item MESSAGES/USER DATA of MVS components, as shown in the following figure.

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                                 CMD Processing              Row 1 to 4 of 20
Command ===> ____________________________________________  SCROLL===> PAGE

Entry Name : MVS_COMPONENTS      Message ID : IEE043I

Enter commands to be executed when resource issues the selected message.
or define this message as status message.

Status . . . ____________   ('?' for selection list)

Pass/Selection Automated Function/'*'
Command Text

_____ ________
MVS WRITELOG START
_____________________________________________________________________________
_____________________________________________________________________________
```

Figure 40. MESSAGES/USER DATA Policy Item for Entry/Type-Pair MVSESA/LOG

Assume that the following messages arrive the first time for one day, while the Job Entry Subsystem is up and running and the recovery automation flag for the MVS component minor resource LOG has not been switched off:

```
IEE043I A SYSTEM LOG DATA SET HAS BEEN QUEUED TO SYSOUT CLASS 1
IEE037D LOG NOT ACTIVE
```

Because IEE043I has been received prior to message IEE037D and the critical threshold that has been defined for message IEE037D has not been exceeded, the command that has been defined for message IEE043I is issued in response to message IEE037D.

Appendix A. Global Variables

You must ensure that the names of any global variables you create do not clash with SA z/OS external or internal global variable names. You should check the following tables before creating any global variables of your own.

Read-Only Variables

There are two different classes of variables, based on the level of access available to the programmer:

Class 1:
Read-only variables. These variables are set by SA z/OS and require at minimum an automation control file reload to be changed.

Class 2:
Read-only variables. These variables are set by SA z/OS CLISTs. They should not be changed except by calling the appropriate CLISTs.

Table 10. Externalized Common Global Variables

Variable Name	Description	Class	Reference
AOF.*clist*.0DEBUG	Contains either a Y or blank. If it contains Y then an intermediate level of debug supported by SA z/OS CLISTs is turned on.	2	
AOF.*clist*.0TRACE	Contains a REXX trace setting to be used by the CLIST *clist*.	2	
AOFAOCCLONE*x*	Where *x* either does not exist (AOFAOCCLONE) or is a value from 1 through 9 or A through Z. The AOFAOCCLONE global variables contain the values specified for the &AOCCLONE. IDs for this system.	1	See the description of the System policy object in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.
AOFCOMPL	Contains YES if initialization is complete.	2	
AOFDEBUG	Contains a REXX trace setting to be used globally.	2	See *IBM Tivoli System Automation for z/OS Planning and Installation*.
AOFINITIALSTARTTYP	Contains the value 'IPL' or 'RECYCLE' depending on whether SA z/OS has been started the first time after an IPL or after a NetView recycle.	1	
AOF_NETWORK_ DOMAIN _ID	Contains the domain name for the NetView that runs network automation as defined in the customization dialog. If not defined, the value of this variable is null.	1	See the description of the System policy object in *IBM Tivoli System Automation for z/OS Defining Automation Policy*.
AOF_PRODLVL	Contains the release level of AOC/MVS or SA z/OS. • For SA OS/390 2.2, the value is V2R2M0. • For SA z/OS 2.3, the value is V2R3M0. • For SA z/OS 3.1, the value is V3R1M0.	1	

Table 10. Externalized Common Global Variables (continued)

Variable Name	Description	Class	Reference
AOFJESPREFX	The command prefix for the primary scheduling subsystem.	1	
AOFSUBSYS	The subsystem name of the primary scheduling subsystem.	1	
AOFSYSNAME	Contains the name of the system.	1	See AOCUPDT in *IBM Tivoli System Automation for z/OS Programmer's Reference*.
AOFSYSTEM	Contains the system type (MVSESA) as defined in the customization dialog.	1	The SYSTEM INFO panel of the customization dialog.

Read/Write Variables

Table 11 lists the common global variables that can be user-defined. You can set them in your startup exit to change the way that SA z/OS behaves. These variables should be set only once for an SA z/OS system. You can enable or disable advanced automation options (AAOs) by changing the settings of the global variables in your CNMSTGEN stylesheet. For example:

```
*****************************************************
* System Automation AAO CGlobals
*****************************************************
COMMON.AOFCNMASK = 290C0D0E0F101518
COMMON.INGREQ_ORIGINATOR = 1
COMMON.AOFRESTARTALWAYS = 0
COMMON.AOFUPDRODM = NO
COMMON.AOFUPDAM = NO
COMMON.AOFSMARTMAT = 0
```

After modifying the exit, an SA z/OS COLD START is required for these changes to take effect.

Table 11. Global Variables to Enable Advanced Automation (CGLOBALS)

Variable	Value	Effect
AOF_AAO_MSG_EHKVAR	**YES**	This indicates that when calling generic routines, the tokens of the triggering message are to be stored in variables EHKVAR0 through EHKVAR9 and EHKVART, if not specified in parameter EHKVAR. YES is the default.
	NO	This indicates that the tokens of the triggering message are not to be stored in EHKVAR variables, if not specified in parameter EHKVAR.
AOF_AAO_MVSTAPEMON	>0	Set this value to represent the number of iterations for INGRTAPE to continue monitoring using MVS commands after the LATE alert has been reached. A non-zero entry will also indicate to use MVS commands for all tape mount monitoring prior to the LATE alert.
	0	INGRTAPE will rely on receipt of the DOMMED message to satisfy any outstanding alerts.
AOF_AAO_RETENTIONPERIOD	0 to 1440	Defines how long (in minutes) SA z/OS should keep the CGLOBALS that are used to keep track of command requests that are received from TWS. The default is 60 minutes.

Table 11. Global Variables to Enable Advanced Automation (CGLOBALS) (continued)

Variable	Value	Effect
AOF_AAO_TRANRERUN	YES	This indicates that a transient job can be rerun within the lifecycle of a particular z/OS, if not specified otherwise in the automation policy for this job.
	NO	This indicates that a transient job will only be run once in the lifecycle of a particular z/OS, if not specified otherwise in the automation policy for this job. NO is the default value.
AOF_AAO_TWS_ERRMSG		This AAO can be used to inhibit the ERRMSG parameter. If set to NON BLANK, it will erase the contents of the ERRMSG parameter.
AOF_AAO_TWS_MAX_WAIT_TIME		Defines the installation default for the maximum wait time for the INGREQ and INGMOVE command. The default is taken when no wait time is specified in the completion information parameter.
AOF_ASSIGN_JOBNAME	**1**	This indicates that SA z/OS will exploit the NetView ″ASSIGN BY JOBNAME″ feature with a higher priority than the "ASSIGN BY MESSAGE ID" feature (priority level 3). This is the default setting.
	0	SA z/OS will exploit the NetView ″ASSIGN BY JOBNAME″ feature with a lower priority than the "ASSIGN BY MESSAGE ID" feature (priority level 4).
AOF_E2E_EAS_PPI	User-defined	PPI receiver ID of the event/automation service to be used to forward events to the end-to-end automation adapter.
AOF_E2E_EVT_RETRY	1 to *n*	Specifies the number of retries, at intervals of one second, that are used to transfer events via PPI TECROUTE to the message adapter of the event/automation service. The events are then forwarded to the end-to-end automation adapter.
AOF_E2E_EXREQ_NETLOG	1	The output to requests received from the end-to-end automation adapter and issued by the primary automation agent, is logged to the NetView log.
	0	The output to those requests is not logged to the NetView log. 0 is the default setting.
AOF_E2E_TKOVR_TIMEOUT	hh:mm:ss	If a hot restart of the automation manager takes longer than the value specified in this variable, the end-to-end automation manager is informed about the outage and has to resynchronize with the first-level automation.
AOF_EMCS_AUTOTASK_ ASSIGNMENT	1	SA z/OS will assign an autotask to extended MCS consoles with a console status of MASTER or ACTIVE
	0	SA z/OS will not assign an autotask to extended MCS consoles with a console status of MASTER or ACTIVE 0 is the default.

Table 11. Global Variables to Enable Advanced Automation (CGLOBALS) (continued)

Variable	Value	Effect
AOF_EMCS_CN_ASSIGNMENT	**1**	SA z/OS will obtain an extended MCS console with a unique name for operator station tasks (OSTs). If an MVS console was obtained for the OST previously, it will be released. 1 is the default setting.
	0	SA z/OS will not obtain an extended MCS console with a unique name for OSTs and the command AOCGETCN will be disabled.
AOFACFINIT	**1**	This indicates that SA z/OS will attempt to proceed with initialization despite error messages such as AOF722I during the processing of the automation control file. 1 is the default setting.
	0	SA z/OS will stop the initialization process upon such errors.
AOFARMQUERYRETRYS	User-defined numeric value.	The number of times AOFPARMQ will be called to query the ARM status of an element after a status of UNKNOWN is returned. If the ARM status does not change to another status before the number of retries is exhausted, SA z/OS will continue processing and assume the element is not ARM-enabled. The default is 10.
AOFARMQUERYWAIT	User-defined numeric value.	The number of seconds to wait between retries as specified in the AOFARMQUERYRETRYS value above. The default is 15.

Table 11. Global Variables to Enable Advanced Automation (CGLOBALS) (continued)

Variable	Value	Effect

AOFCNMASK		The characters that are used in determining unique console names can be tailored by updating the common global variable AOFCNMASK. This global is used as a hex mask to extract characters from the following string when generating unique console names with command AOCGETCN:
		`left(opid(),8)||right(opid(),8),`
		`||left(aofsysname,4)||right(aofsysname,4),`
		`||left(applid(),8)||right(applid(),8),`
		`||'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789$&#155;#@_!?'`
		Where
		• opid() is a function that returns the OST task name
		• aofsysname is a common global that stores the system name
		• applid() is a function that returns VTAM LU name
		The default for AOFCNMASK is 290C0D0E0F101718. X'29' selects character A in position 41, X'0C' through X'10' selects the last five characters of the opid in positions 12 to 16, X'17' and X'18' select the last two characters of the sysname in positions 23 and 24.
		If AOFCNMASK is null, AOCGETCN will attempt to obtain a unique extended MCS console after a 1 minute interval, followed by a two minute interval and so forth for a maximum of 5 passes (15 minutes elapsed from the initial invocation of the command).
		For example, with
		`AOFCNMASK: 2A01020304051718`
		X'2A' selects character B in position 42, X'01' through X'05' selects the first five characters of the opid in positions 1 to 5, X'17' and X'18' select the last two characters of the sysname in positions 23 and 24.

AOFDEFAULT_TARGET	User-defined	Sets a default for the TARGET parameter for all commands where this parameter is used.
AOFDESCA	0100001000001000	Descriptor code for action messages
AOFDESCD	0100001000001000	Descriptor code for decision messages
AOFDESCE	0010001000001000	Descriptor code for eventual action messages
AOFDESCI	0000011000001000	Descriptor code for informational messages
AOFDESCW	1000001000001000	Descriptor code for wait messages
AOFEXPLAIN_USER	User-defined	The EXPLAIN command accepts this variable to include help support for customer installation supplied terms. It can hold one or more pairs of ***term/help panel*** specifications separated by a blank. If the specified status in the EXPLAIN command is not a valid SA z/OS status, the command routine will check whether it is an installation defined term. If so, the associated help panel is displayed.

Table 11. Global Variables to Enable Advanced Automation (CGLOBALS) (continued)

Variable	Value	Effect
AOFINITREPLY	**hh:mm:ss**	The initial reply AOF603D is issued and automatically responded after hh:mm:ss. **00:02:00** (2 minutes) is the default setting.
	0	The initial reply AOF603D will not be issued and automation continues with the default start without asking the operator.
AOF_INIT_MCSFLAG	User-defined valid value	This variable contains the MCSFLAG that is used for WTOs and WTORs that are issued by SA z/OS during initialization. The default is '00000000'.
AOF_INIT_ROUTCDE	User-defined valid value	This variable contains the ROUTCDE (routing code) that is used for WTOs and WTORs that are issued by SA z/OS during initialization. The default is '01000000'.
AOF_INIT_SYSCONID	User-defined valid value	This variable contains the SYSCONID that is used for WTOs and WTORs that are issued by SA z/OS during initialization. The default is blank.
AOFLOCALHOLD	**0**	INGNTFY and SA z/OS initialization will execute the SETHOLD AUTO command on the notify operator. 0 is the default setting.
	1	SETHOLD must be manually invoked.
AOFMATLISTING	**0**	Setting this variable means that the NetView automation table listing is not placed in the DSILIST data set at NetView automation table load time.
AOFOPCCMDMSG	**0**	OPCAMOD will only produce messages generated by INGOPC. 0 is the default setting.
	1	OPCAMOD will produce EVJ011I, EVJ412I, EVJ420I, and EVJ423I messages.
AOFPAUSE	**0** to 5	This is the number of seconds that SA z/OS will allow for applications that have shut down to be cleared by MVS, in addition to their termination delay. As the AOFPAUSE value is applied to all applications it should be kept small. AOFPAUSE may be useful on a slow machine, where allowing an extra second or two before SA z/OS checks if the application has been cleared could avoid the need to use a termination delay timer. No matter how AOFPAUSE is set, the application status will not be updated to AUTODOWN or CTLDOWN until SA z/OS is sure that the application has been cleared from the system by MVS. 0 is the default setting.

Table 11. Global Variables to Enable Advanced Automation (CGLOBALS) (continued)

Variable	Value	Effect
AOFRESTARTALWAYS	1	An application that has been shut down normally, outside the control of SA z/OS, with RESTARTOPT=ALWAYS, will be restarted regardless of whether or not it has reached its critical error threshold.
	0	An application that has been shut down normally, outside the control of SA z/OS, with RESTARTOPT=ALWAYS, will *not* be restarted if it has reached its critical error threshold. 0 is the default setting.
AOFRMTCMDWAIT	See NetView RMTCMD	Contains the installation wait time when RMTCMD is used for communication. **60 seconds** is the default setting for RMTCMD.
AOFRPCWAIT	0 to *n*	This is the number of seconds that SA z/OS will wait for command responses from other systems in the sysplex. **10** is the default setting.
AOFSENDALERT	**Yes** or No	This defines whether NetView alert forwarding (YES) or the command handler (NO) is used to forward data to the focal point. **Yes** is the default setting.
AOFSERXINT	**1**	The exit AOFEXINT is processed under the BASEOPER automation operator under the initialization process. This is the default.
	0	The exit AOFEXINT execution is serialized within the initialization process.
AOF_SET_AVM_RESTART_EXIT	**1**	SA z/OS will set the AVM restart exit during the initialization of the automation environment. **1** is the default.
	0	The AVM restart exit needs to be set in the SYS1.PARMLIB PROGxx member. Please refer to *IBM Tivoli System Automation for z/OS Planning and Installation* for more information.
AOFSHUTDELAY	0 to 59	This is the number of minutes that SA z/OS will wait for a termination message before continuing the shutdown process. Any values outside this range are treated as 0. With a setting of 0, message AOF745E will not be issued. **0** is the default setting.

Table 11. Global Variables to Enable Advanced Automation (CGLOBALS) (continued)

Variable	Value	Effect
AOFSMARTMAT	0	The SA z/OS Agent is disabled from refreshing ATs. The AT fragment INGMSG02 is included when SA z/OS initially loads INGMSG01. Note that INGMSG02 is no longer shipped. You therefore need to change the AT® scope to ENTERPRISE and then run a SysOps build with MODIFIED in the TYPE build option field.
	1	The SA z/OS Agent is enabled to refresh ATs when an INGAMS REFRESH is issued. The AT fragment built by the customization dialog is *not* loaded; INGMSG02 is used instead. Note that INGMSG02 is no longer shipped. You therefore need to change the AT scope to ENTERPRISE and then run a SysOps build with MODIFIED in the TYPE build option field. The ATs will be loaded after a successful test load. This will allow the agent to inform the AM about a load problem of the AT. The agent may inform the AM of an AT load failure, thus stopping the configuration refresh.
	2	The SA z/OS Agent is enabled to load the AT that is generated by the customization dialog and to refresh ATs when an INGAMS REFRESH is issued. The AT that is built by the customization dialog is dynamically loaded into storage as the INGMSG02 fragment. The ATs will be loaded after a successful test load. This will allow the agent to inform the AM about a load problem of the AT. The agent may inform the AM of an AT load failure, thus stopping the configuration refresh. This is the default value.
AOFSPOOLFULLCMD	1	SA z/OS will not execute the Spool recovery passes more than once. Message AOF2941I will be issued if the SPOOLFULL condition persists.
	0	SA z/OS will re-execute the Spool recovery commands. 0 is the default setting.
AOFSPOOLSHORTCMD	1	SA z/OS will not execute the Spool recovery passes more than once. Message AOF2941I will be issued if the SPOOLSHORT condition persists.
	0	SA z/OS will re-execute the Spool recovery commands. 0 is the default setting.
AOFSTATUSCMDSEL	0	Issue all status commands or replies that are associated with the new status, without respect to any specified selection values. No thresholds are checked for the minor resource *subsystem.status* to derive selection criteria or prevent the issuing of commands or replies if critical thresholds are exceeded. If AOFSTATUSCMDSEL is not set, or it is set to a value other than 0, only commands or replies with a given selection criterion such as starttype or stoptype are issued.

Table 11. Global Variables to Enable Advanced Automation (CGLOBALS) (continued)

Variable	Value	Effect
AOFUPDAM	Yes or **No**	This controls whether updates are made in the automation manager. **No** is the default setting.
AOFUPDRODM	Yes or **No**	This controls whether updates are made in RODM and must be set to the same value for each system within a sysplex. **No** is the default setting.
AOFUSSWAIT	1 to *n*	This is the number of seconds SA z/OS waits for the completion of a user-defined z/OS UNIX monitoring routine (specified in the z/OS UNIX Control Specification panel) until it gets a timeout. When the timeout occurs, SA z/OS does no longer wait for a response from the monitoring routine and sends a SIGKILL to the monitoring routine. **10** is the default setting.
INGIMS_CORRWAIT	User-defined numeric value	Number of seconds that INGIMS will wait for output from an IMS command. If not specified, INGIMS will use the default CORRWAIT (CCDEF) value.
INGOPC_MULTIPLIER	1 to *n*	This is used in conjunction with AOFRMTCMDWAIT and AOFRPCWAIT to determine how long to wait before giving up.
INGREQ_ORIGINATOR	1	Indicates that SA z/OS assigns individual originator IDs for each operator issuing an INGREQ command.
	0	All operators are grouped under originator ID OPERATOR. 0 is the default setting.

Parameter Defaults for Commands

Table 12. Global Variables That Define the Installation Defaults for Specific Commands

Variable Name	Description	Reference [1]
AOFSETSTATEOVERRIDE	Sets the default OVERRIDE value for the SETSTATE command.	SETSTATE
AOFSETSTATESCOPE	Allows you to override the predefined default for the SCOPE parameter of the SETSTATE command.	SETSTATE
AOFSETSTATESTART	Allows you to override the predefined default for the START parameter of the SETSTATE command.	SETSTATE
DISPEVT_WAIT	Sets the WAIT parameter of the DISPEVT command to the specified value.	DISPEVT
DISPEVTS_WAIT	Sets the WAIT parameter of the DISPEVTS command to the specified value.	DISPEVTS
DISPTRG_WAIT	Sets the WAIT parameter of the DISPTRG command to the specified value.	DISPTRG
INGAUTO_INTERVAL	Sets the default for the INTERVAL parameter of the INGAUTO command.	INGAUTO

Table 12. Global Variables That Define the Installation Defaults for Specific Commands (continued)

Variable Name	Description	Reference [1]
INGEVENT_WAIT	Sets the WAIT parameter of the INGEVENT command to the specified value. The parameter specifies whether or not to wait until the request is complete.	INGEVENT
INGGROUP_WAIT	Sets the WAIT parameter of the INGGROUP command to the specified value. The parameter specifies whether or not to wait until the request is complete.	INGGROUP
INGHIST_MAX	Sets the MAX parameter of the INGHIST command to the specified value.	INGHIST
INGINFO_WAIT	Sets the WAIT parameter of the INGINFO command to the specified value.	INGINFO
INGLIST_WAIT	Sets the WAIT parameter of the INGLIST command to the specified value.	INGLIST
INGMOVE_WAIT	Sets the WAIT parameter of the INGMOVE command to the specified value.	INGMOVE
INGRELS_SHOW	Sets the SHOW parameter of the INGRELS command to the specified value.	INGRELS
INGRELS_WAIT	Sets the WAIT parameter of the INGRELS command to the specified value.	INGRELS
INGREQ_EXPIRE	Sets the default EXPIRE parameter of the INGREQ command to the specified value.	INGREQ
INGREQ_INTERRUPT	Sets the default INTERRUPT parameter of the INGREQ command to the specified value. The parameter specifies whether or not the automation manager should wait until the resource has reached its UP state, but the resource is still in the startup phase when the higher priority stop request is given.	INGREQ
INGREQ_OVERRIDE	Sets the default OVERRIDE parameter of the INGREQ command to the specified value.	INGREQ
INGREQ_PRECHECK	Sets the default PRECHECK parameter of the INGREQ command to the specified value.	INGREQ
INGREQ_PRI	Sets the default priority (PRI parameter) of the INGREQ command to the specified value.	INGREQ
INGREQ_PRI.E2EMGR	Specifies the priority that incoming requests from the end-to-end automation manager are executed at. Default: LOW	INGREQ
INGREQ_REMOVE	Sets the default value for the REMOVE parameter of the INGREQ command to the specified value. It the resource reaches the specified status (condition), the request is automatically removed.	INGREQ
INGREQ_REMOVE.START	Sets the default value for the REMOVE parameter of the INGREQ START command. If not specified the value set by INGREQ_REMOVE will be used.	INGREQ
INGREQ_REMOVE.STOP	Sets the default value for the REMOVE parameter of the INGREQ STOP command. If not specified the value set by INGREQ_REMOVE will be used.	INGREQ
INGREQ_RESTART	Sets the default for the RESTART parameter of the INGREQ command when shutting down the resource.	INGREQ
INGREQ_SCOPE	Sets the SCOPE parameter of the INGREQ command to the specified value.	INGREQ
INGREQ_SOURCE	Sets the default SOURCE parameter of the INGREQ command to the specified value. The parameter specifies the originator of the request.	INGREQ

Table 12. Global Variables That Define the Installation Defaults for Specific Commands (continued)

Variable Name	Description	Reference [1]
INGREQ_TIMEOUT	Sets the interval in minutes used to check for the INGREQ command used to check whether the request has been successfully completed, and whether to send a message or cancel the request if it has not been satisfied after that time.	INGREQ
INGREQ_TYPE	Sets the default startup/shutdown type (TYPE parameter) of the INGREQ command to the specified value.	INGREQ
INGREQ_VERIFY	Sets the default VERIFY parameter of the INGREQ command to the specified value.	INGREQ
INGREQ_WAIT	Sets the WAIT parameter of the INGREQ command to the specified value.	INGREQ
INGSCHED_WAIT	Sets the WAIT parameter of the INGSCHED command to the specified value. The parameter specifies whether or not to wait until the request is complete.	INGSCHED
INGSET_VERIFY	Sets the default VERIFY parameter of the INGSET command to the specified value.	INGSET
INGSET_WAIT	Sets the WAIT parameter of the INGSET command to the specified value. The parameter specifies whether or not to wait until the request is complete.	INGSET
INGTRIG_WAIT	Sets the WAIT parameter of the INGTRIG command to the specified value.	INGTRIG
INGVOTE_EXCLUDE	Sets the EXCLUDE parameter of the INGVOTE command to the specified value. The parameter specifies the resource types (for example SVP or GRP) to be excluded when showing all requests. Resources of that type are filtered out.	INGVOTE
INGVOTE_STATUS	Sets the STATUS parameter of the INGVOTE command to the specified value. The parameter specifies which requests should be displayed: winning, losing or all.	INGVOTE
INGVOTE_VERIFY	Sets the default VERIFY parameter of the INGVOTE command to the specified value.	INGVOTE
INGVOTE_WAIT	Sets the WAIT parameter of the INGVOTE command to the specified value.	INGVOTE

1. See the specified command in *IBM Tivoli System Automation for z/OS Operator's Commands*.

Appendix B. Customizing the Status Display Facility

Overview of the Status Display Facility

This appendix explains how to customize the Status Display Facility (SDF) panels, descriptors, and operations.

How the Status Display Facility Works

The SA z/OS Status Display Facility (SDF) uses colors and highlighting to represent subsystem resource states. Typically, a subsystem shown in green on the SDF status panel indicates it is up, while red indicates a subsystem in a stopped or problem state. SDF can be tailored to present the status of system components in a hierarchical manner.

Note: SDF works only with MVS systems and resources.

Types of SDF Panels

Figure 41 on page 248 shows several SDF screens for system CHI01. This figure shows the main types of panels used in SDF:

- The *root component*
- The *status component*
- The *detail status display*

In addition to these panel types, you can create other types of panels according to your system requirements and the applications you are monitoring.

Note: All SDF panels must contain 24 rows and 80 columns. Because SDF uses only the display's default screen size, the default size must be defined as 24 x 80.

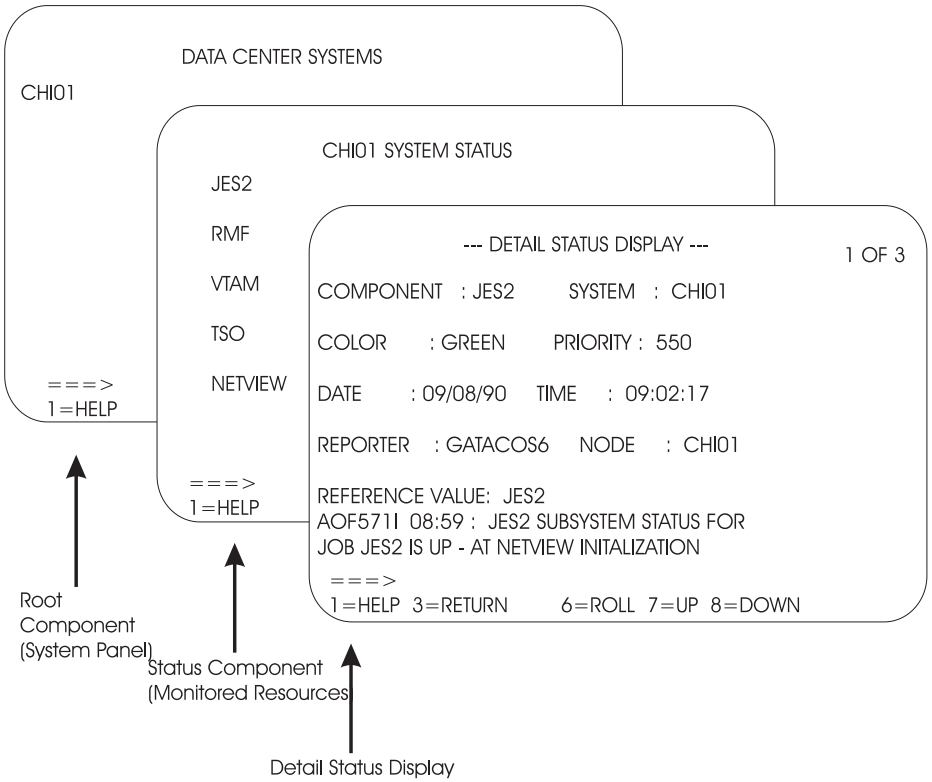


Figure 41. Example SDF Panels

Root Component

The root component is typically an element appearing on the first screen displayed when SDF is started. In Figure 41, the CHI01 system is the root component.

Status Component

Resources monitored by SDF are called *status components*. In Figure 41, system CHI01 has JES2, RMF, VTAM, TSO, and NetView status components, as shown on the CHI01 System Status panel. The status component panel displays all monitored resources in a system. Each monitored resource is shown in the color of its current status. For example, JES2 is shown in green if it is up.

Detail Status Display

A detail status display is built from information in a status descriptor (see "Status Descriptors"). This panel is displayed by tabbing to the appropriate resource on the status component panel and pressing the detail PF key. Each status component can have one or more status descriptors, or detail records, associated with it.

Figure 41 shows an example detail status display for a JES2 status descriptor. The `1 of 3` on the panel indicates that JES2 currently has three status descriptors, and therefore three detail status displays, associated with it.

Status Descriptors

A *status descriptor* is a detailed record of information about a resource status. In its raw form, a status descriptor is a multiline SA z/OS message containing information such as:

- Root component and status component to which the status descriptor applies

- Priority, color, and highlighting associated with the status descriptor (see "How Status Descriptors Affect SDF" on page 251 for more information)
- Date and time the status descriptor was generated
- Actual resource status information; for example, an SA z/OS message indicating the resource is up

SDF uses information in a status descriptor to generate a detail status display (see "Detail Status Display" on page 248). You do not usually look directly at a status descriptor; rather, you look at portions of it through a detail status display. For example, in Figure 41 on page 248, the detail status display presents information from a status descriptor for status component JES2. The `1 of 3` on the panel indicates that JES2 currently has three status descriptors associated with it.

SDF generates, displays, and deletes status descriptors.

SDF Tree Structures

SDF uses *tree structures* to set up the hierarchy of monitored resources displayed on SDF status panels. An SDF tree structure always starts with the system name as the root node and has a level number of one. Tree structure levels subordinate to the root node are the monitored resources. The level numbers of these resources reflect their dependency on each other.

You define SDF tree structures in NetView DSIPARM data set member AOFTREE.

Figure 42 on page 250 shows an example SDF tree structure. Following the tree structure definition statements is a diagram showing how these statements result in a tree structure.

```
1 SY1
  2 SYSTEM
    3 WTOR
    3 APPLIC
      4 AOFAPPL
        5 AOFSSI
      4 JES
      4 VTAM
    3 TSO
    3 RMF
  2 GATEWAY
```

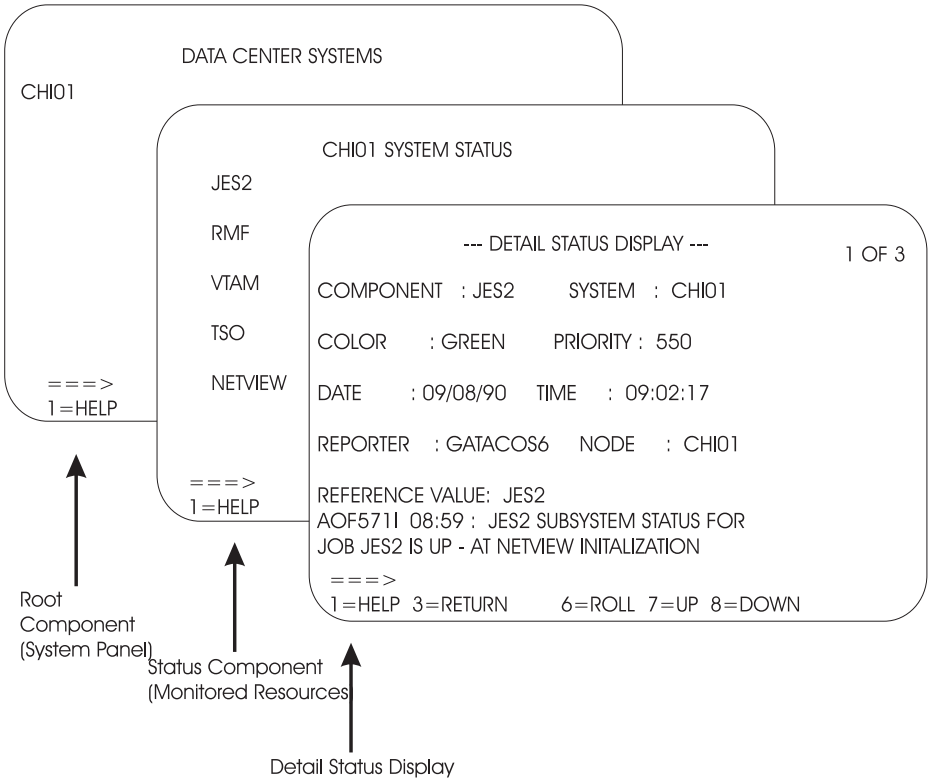


Figure 42. Example SDF Tree Structure

SA z/OS supplies a sample SDF tree structure in the SA z/OS sample library. This tree structure is referenced by a %INCLUDE statement in member AOFTREE in the NetView DSIPARM data set. You can customize this sample tree structure to meet your requirements. This order of dependency does *not* have to be the same as that used for system startup or shutdown using SA z/OS.System symbols are supported for the tree structure. This can help reduce both customization work and errors.

For example, using the tree structure in Figure 42, if there is a problem with TSO, it is not desirable to also change the VTAM status color, because VTAM is not having any problems. In contrast, in the SA z/OS startup and shutdown procedures, TSO is dependent on VTAM.

More details on SDF tree structure definitions are in “Step 1: Defining SDF Hierarchy” on page 257.

How Status Descriptors Affect SDF

Status descriptors are the main units of information SDF uses. The information in status descriptors determines how your SDF status displays look at any point in time. This section explains how SDF uses status descriptors.

Priority and Color Assignments

Status descriptors are assigned both a priority number and a color. These color and priority assignments determine the colors in which status components are displayed. In SDF, a lower number indicates a higher priority. Status descriptors are connected to the status component in ascending order of priority.

Color and priority assignments for status descriptors are defined in two places:

- In the PRIORITY parameter in the AOFINIT member of the NetView DSIPARM data set. This parameter defines initial priority and color assignments used for status descriptors. The values defined in AOFINIT are used if no further customization is done to priority and color assignments. The default priority ranges and colors used in AOFINIT are:

Priority Range	Color
001 to 199	Red
200 to 299	Pink
300 to 399	Yellow
400 to 499	Turquoise
500 to 599	Green
600 to 699	Blue

  White is used as the default status descriptor color (the DCOLOR parameter in member AOFINIT, described in *IBM Tivoli System Automation for z/OS Programmer's Reference*) and as the default color for a status component without a tree structure entry (the ERRCOLOR parameter in member AOFINIT, described in *IBM Tivoli System Automation for z/OS Programmer's Reference*). For more information on the PRIORITY parameter, see *IBM Tivoli System Automation for z/OS Programmer's Reference*.
- In the SDF definitions in the Status Details policy object. These entries define colors, highlighting, and priorities used for particular resource statuses. Color and priority assignments defined in the customization dialog can be used to override assignments in the AOFINIT member.

  Note: Some of the resource statuses that appear in SDF displays do not directly correspond to resource statuses used in the automation status file.

  IBM Tivoli System Automation for z/OS User's Guide shows the default resource status types, colors, highlighting, and priorities provided with SA z/OS. These settings define to SA z/OS the parameters used when adding status descriptors to SDF.

  For more information on the SDF Status Details definition, see "Step 4: (Optional) Defining SDF in the Customization Dialog" on page 262.

Chaining of Status Descriptors to Status Components

A resource status change causes a status descriptor to be generated. SDF adds this status descriptor to a chain of status descriptors. Chained status descriptors determine the status and color of status components. The highest-priority status descriptor in a chain determines the initial color in which the status component is

displayed. The underlying chained priority numbers determine the color in which successive detail status displays will be shown.

Status descriptors are chained off each level of status component in a tree structure. Status descriptors chained to lower-level status components are also chained to a higher-level status component, again in order of priority. Status descriptors are also chained off the root component. These status descriptors are all the status descriptors that currently exist at all levels of the tree structure.

For example, Figure 43 shows status descriptors currently generated for system SY1. The priority for each status descriptor is shown by a number.

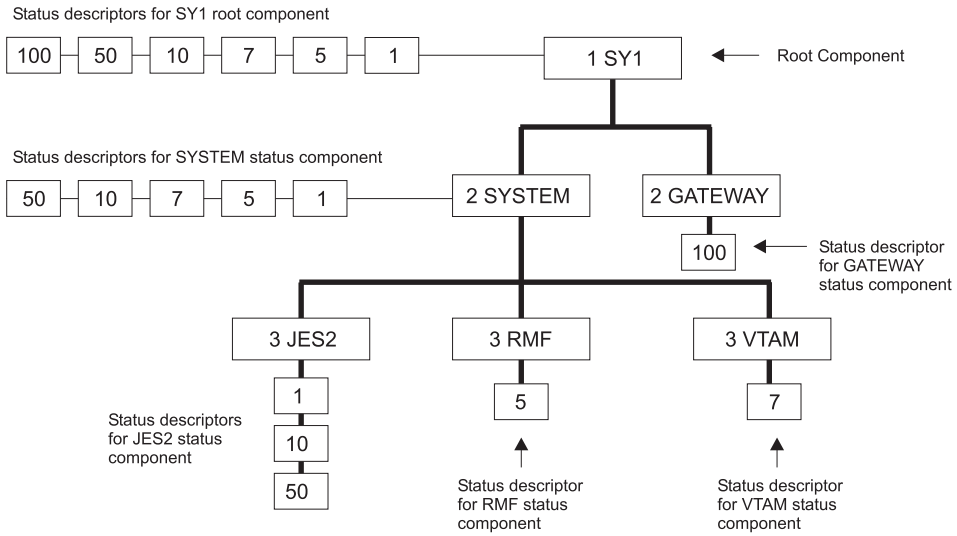


Figure 43. Status Descriptors Chained to Status Components

The status components at the lowest level in this tree structure, JES2, RMF, and VTAM, have status descriptors chained off them. Status component JES2 has three status descriptors chained, with priorities 1, 10, and 50. Because 1 is the highest priority, the status descriptor with priority 1 is organized first in the chain. This highest-priority status descriptor determines the color in which JES2 is displayed on the status panel. If an operator uses the detail PF key to view detail status displays for JES2, the information contained in the status descriptor with priority 1 will be displayed first, then the detail status display for the status descriptor with priority 10, and so on.

At the SYSTEM status component level in the tree structure, all status descriptors from the lower-level status components are also chained. Because the status descriptors chained to RMF and VTAM have higher priorities than the priority 10 and 50 status descriptors for JES2, they are organized after the priority 1 status descriptor in the chain. An operator using the detail PF key at the SYSTEM level could view five detail status displays, ranging from priority 1 to priority 50.

Similarly, at the SY1 level in the tree structure, all status descriptors chained to all status components in the tree structure are chained in order of priority. An operator using the detail PF key at the SY1 level could view six detail status displays, ranging from priority 1 to priority 100.

If a status component has multiple status descriptors with equal priorities, the status descriptors are chained off the status component in order of arrival time.

When a status descriptor no longer accurately reflects the actual status of a resource, SDF automatically deletes it from status descriptor chains. As an example of how priority determines order of status descriptors, suppose two status descriptors currently exist for status component JES2. If there are two status descriptors for JES2 with priorities of 120 and 140, the status descriptor with priority 120 is displayed first. In both cases, JES displays in red on the SDF status panel.

In SA z/OS, all statuses are defined in the automation control file. When an automation event occurs, the SA z/OS AOCUPDT common routine scans the automation control file for the SDF entry for that status. SA z/OS issues a request to add the status using the information from the automation control file.

For example, suppose subsystem RMF, shown on the example SDF panels in Figure 41 on page 248, is set to a STOPPING state. The SA z/OS AOCUPDT common routine scans the automation control file for the STOPPING state entry for SDF and generates a status descriptor, specifying a priority of 330. SDF adds the status descriptor to the RMF status component. RMF appears as yellow and blinking on the status panel. Once RMF is in a stopped state, the AOCUPDT common routine scans the automation control file for the STOPPED state SDF entry and generates a status descriptor with priority 130. SDF adds this new status descriptor to the RMF status component. Now, RMF appears in red on the SDF status panel.

Propagating Status Descriptors Upward and Downward in a Tree Structure

Based on the order of dependencies defined in a tree structure, status descriptors can be *propagated* upward or downward to status components in a tree structure. This propagation of status descriptors affects the color in which status components are displayed, as well as the detail status displays operators can view by using the detail PF key on a particular status component.

Propagation of status upward and downward in a tree structure is defined by the PROPUP and PROPDOWN parameter in the AOFINIT member (see *IBM Tivoli System Automation for z/OS Programmer's Reference* for descriptions).

The SA z/OS-provided defaults for status propagation in the AOFINIT member are to propagate status upward (PROPUP=YES) but not downward (PROPDOWN=NO).

When status is propagated upward in a tree structure, if a status descriptor is added or deleted at a lower level in the tree structure, it is also added or deleted from the cumulative chain of status descriptors at a higher-level node in the tree structure.

Propagation of status upward in a tree structure consolidates the status of all monitored resources in the system at the root node. In this way, the color of the root node reflects the most important or critical status in a computer operations center. For example, in Figure 42 on page 250, any color changes for AOFSSI are reflected in AOFAPPL, APPLIC, SYSTEM, and SY1, if SDF propagates status changes upward in the tree structure. In Figure 41 on page 248, if all monitored resources are green, the root node CHI01 on the Data Center Systems panel is also shown in green.

When status is propagated downward in a tree structure, if a status change occurs at a higher level in a tree structure, the changes are sent downward in the tree

structure. This propagating downward could cause status descriptors at lower levels in the tree structure to be added or deleted.

Propagating status downward can be useful when an entire system is down. In such a case, you want SDF status panels to accurately reflect the system status. You do not want status components lower in the tree structure to retain previously generated status descriptors indicating that the components are up and running, because these status descriptors do not accurately reflect the status of the components. You can configure your SDF implementation to propagate status downward, and remove all status descriptors from all status components in a tree structure. If an operator tries displaying detailed status about any of the status components lower in the tree structure, they receive "NO DETAIL INFO AVAILABLE" messages. The empty chain color, defined by the EMPTYCOLOR parameter in member AOFINIT with a default color of blue, is also used to indicate that no detail information is available. See *IBM Tivoli System Automation for z/OS Programmer's Reference* for the EMPTYCOLOR description.

How SDF Helps Operations to Focus on Specific Problems

SDF structure and processing allows the program identifying a problem to be concerned only with the specific problem.

For example, suppose an application program detects a warning message for status component JES on CHI01. The following processing steps occur:

1. The application program issues a request to SDF to add a status descriptor for JES.
2. The status entry for JES on system CHI01 now indicates there is a problem with JES. If the SDF is configured to propagate status up the hierarchical tree structure, the status for system CHI01 also reflects the problem state. See *IBM Tivoli System Automation for z/OS Programmer's Reference* for details on the PROPUP SDF initialization parameter.
3. Now, suppose another more serious problem occurs. The application program which detects this new problem issues another request to SDF to add a status descriptor having a lower priority number than the status descriptor for the first problem.
4. Because status descriptors are chained in order of priority, the JES status now reflects the status descriptor color of the more serious problem.
5. When the more serious problem is resolved, the application program detecting the problem resolution issues a request to SDF to remove the status descriptor for this problem from the chain of JES status descriptors.
6. The status panel is updated to reflect the first problem.

How SDF Panels Are Defined

All SDF status panels, apart from detail status display panels, are defined in the AOFPNLS member of the NetView DSIPARM data set.

Member AOFPNLS can contain either one or both of the following:

- %INCLUDE statements referencing other NetView DSIPARM members containing definitions of panels. The %INCLUDE statement causes the named panel definition member to be loaded. This is the recommended method, and the method used in the SA z/OS-provided version of AOFPNLS. System symbols are supported for the %INCLUDE statements. This can help reduce both customization work and errors.
- Panel structure definitions for all SDF panels.

Panel members defined or referenced in AOFPNLS are loaded into system memory, and may be deleted, replaced, or temporarily made resident using the SDFPANEL command (see *IBM Tivoli System Automation for z/OS Programmer's Reference* for command description).

Panels that are to be dynamically loaded as needed (see "Dynamically Loading Tree Structure and Panel Definition Members") must be defined in a NetView DSIPARM member having the same member name as the panel itself.

It is recommended that you include only frequently used panels in AOFPNLS, to conserve system memory. Other panels can be dynamically loaded when needed, either by pressing a SDF function key or by using the SCREEN command.

Note: Dynamic refresh will only work with panels defined in AOFPNLS.

SDF internally formats and builds detail status display panels from the information in a status descriptor. You do not have to define and format detail status display panels. Status components defined in the panel definitions must also be defined in the corresponding tree structure. However, not all status components defined in the tree structure require a corresponding entry on the SDF status panel. For example, in Figure 42 on page 250, the APPLIC status component is only a pseudo-entry and may not actually be displayed on any SDF status display panel.

SDF status panels can be customized to reflect any environment. For example, you can define a panel to show the status of all JES subsystems on all processors in a computer operations center. The JES operator can view the panel to determine the status of any JES subsystem in the complex.

For detailed information on defining SDF panels, see "Step 2: Defining SDF Panels" on page 259.

Dynamically Loading Tree Structure and Panel Definition Members

Using %INCLUDE statements in the main SDF tree structure and panel definition members allows you to dynamically load tree structure and panel definition members without restarting SDF (see *IBM Tivoli System Automation for z/OS Programmer's Reference*). The SDFTREE command loads a tree structure definition member. The SDFPANEL command loads a panel definition member. You can dynamically reload members AOFTREE and AOFPNLS themselves.

Using SDF for Multiple Systems

You can configure SDF so that multiple systems in an automation network can forward their resource status information to the SDF on the focal point system. In a multiple-system environment, the following must be defined:

- The tree structure for each system must be defined in the AOFTREE member of NetView DSIPARM on the focal point system SDF. The root name must be unique for each system tree structure.
- For target system SDF status update to occur on a focal point SDF, SA z/OS focal point services must already be implemented.

Because each root name must be unique in a multiple-system environment, any status component on any system defined to the focal point SDF can be uniquely addressed by prefixing the status component with the root component name:

```
ROOT_COMPONENT.STATUS_COMPONENT
```

For example:

```
SY1.JES2
```

Similarly, any SDF status descriptors forwarded from the target system to the focal point SDF are prefixed with the root name of the target system by SA z/OS routines.

SDF Components

SDF consists of the following components:

Table 13. SDF Components

Name	Type	Purpose
AOFTDDF	Task	Initializes SDF and maintains the status database. This initialization is an automated function.
SDF	Command	Starts an SDF operator session.
SDFTREE	Command	Dynamically loads or deletes an SDF tree structure definition member from the NetView DSIPARM data set.
SDFPANEL	Command	Dynamically loads or deletes an SDF panel definition member from the NetView DSIPARM data set.
AOFINIT	Input file	Contains SDF initialization parameters defined with the statements described in *IBM Tivoli System Automation for z/OS Programmer's Reference*. AOFINIT is in the NetView DSIPARM data set.
AOFTREE	Input file	Contains tree structures described in *IBM Tivoli System Automation for z/OS Programmer's Reference*. This member usually consists of a list of %INCLUDE statements referencing other members containing tree structures. AOFTREE is in the NetView DSIPARM data set.
AOFPNLS	Input file	Contains SDF panel parameters defined by the statements described in "Step 2: Defining SDF Panels" on page 259. This member usually consists of a list of %INCLUDE statements referencing other members containing panel definitions. AOFPNLS is in the NetView DSIPARM data set.
panel_name	Input file	A DSIPARM member containing the definition of one or more SDF panels or %INCLUDE statements identifying other DSIPARM panel definition members. It is highly recommended that panel definition members contain the definition of a single panel having the same name as the member.
tree_name	Input file	A DSIPARM member containing the definition of one or more tree structures. It is highly recommended that tree definition members contain the definition of a single tree having the same root component name as the member name.

How the SDF Task Is Started and Stopped

During SA z/OS initialization, the AOFTDDF task loads members defining panel format, panel flow, and tree structures. Member AOFINIT defines parameters common to all SDF panels and basic initialization specifications, such as screen size, default PF keys, and the initial screen displayed when a SDF session is started. These AOFINIT parameters are described in *IBM Tivoli System Automation for z/OS Programmer's Reference*.

Starting the SDF Task

In SA z/OS code, the AOFTDDF task is started by the following command:

```
START TASK=AOFTDDF
```

Stopping the SDF Task

In SA z/OS code, the AOFTDDF task is stopped by the following command:

```
STOP TASK=AOFTDDF
```

Note: When SDF is restarted, all existing SDF status descriptors are lost, as they are kept only in memory.

SDF Definition Process

Use the following procedure to define the panels displayed in an SDF session. Details on each step are provided later in this chapter and in *IBM Tivoli System Automation for z/OS Programmer's Reference*.

1. Define the hierarchy of monitored resources used for your SDF panels, using tree structure statements in NetView DSIPARM data set members. These tree structure definition members should be referenced by %INCLUDE statements in the main SDF tree structure definition member, AOFTREE, in the NetView DSIPARM data set. See *IBM Tivoli System Automation for z/OS Programmer's Reference* for details.
2. Define SDF status panels using panel definition statements in NetView DSIPARM data set members. Panels can either be automatically loaded when SDF starts, or dynamically loaded using the SDFPANEL command. For panels to be automatically loaded, add a %INCLUDE statement specifying the panel definition member to the main panel definition member, AOFPNLS, in the NetView DSIPARM data set. See "Step 2: Defining SDF Panels" on page 259 for details.

   Define and customize SDF status panels in the following general order:
   a. Root panel
   b. Status component panel for each entry on the root panel
   c. Any other customized status panels.
3. Customize the SDF initialization parameters in NetView DSIPARM member AOFINIT, if necessary (optional), or use defaults. See *IBM Tivoli System Automation for z/OS Programmer's Reference* for detailed descriptions of SDF initialization parameters. Using defaults is recommended.
4. Define SDF resource status, color, highlight and priority values using the customization dialog to edit the SDF Status Display policy object, or use defaults. This step is optional. See *IBM Tivoli System Automation for z/OS Defining Automation Policy* for the description of the Status Display policy object. Using defaults is recommended.

Notes:

1. Resources that SA z/OS is not currently automating are not displayed on SDF panels.
2. To display the status of multiple systems and forward status from target systems to SDF on a focal point system, SA z/OS focal point services must already be implemented. See *IBM Tivoli System Automation for z/OS Defining Automation Policy* for details on configuring focal point services.

Step 1: Defining SDF Hierarchy

Member AOFTREE in the NetView DSIPARM data set contains a set of definitions that define the propagation hierarchy for status color changes. When the status

changes for a component, the corresponding color change is propagated up or down the tree to the next higher or lower level component. The level is determined by the level number assigned to each component. The type of propagation is determined either by the entry in the AOFINIT member or by individual requests to add a status descriptor to a status component.

Note: SA z/OS does not use this SDF hierarchy for subsystem shutdown or startup procedures. Instead, SA z/OS uses subsystem entries defined in the automation policy to determine startup and shutdown relationships and hierarchies.

Tree Structure Definitions

AOFTREE contains tree structure definitions. To define tree structures, you can:

- Use %INCLUDE statements that reference other members containing definitions for specific tree structures. This is the recommended method, and the method used in the SA z/OS-provided version of AOFTREE.

  On the %INCLUDE statement, the name of the referenced member must be enclosed in parentheses.
- Place all tree structure definitions in AOFTREE.
- Use a combination of both.

System symbols are supported wherever they are used in the AOFTREE, AOFINIT and AOFPNLS members. This can help reduce both customization work and errors.

Figure 44 shows a typical tree structure definition:

```
1 SY1
  2 APPLIC
     3 AOFAPPL
       4 AOFSSI
     3 JES
     3 VTAM
     3 TSO
     3 RMF
  2 GATEWAY
```

Figure 44. Example Tree Structure Definition

In this tree structure, SY1 is the root component. This definition is in a separate member, named SY1. It is referenced by the following statement in the AOFTREE member:

```
%INCLUDE(SY1TREE)
```

Loading Tree Structures: All tree structures need not be loaded during initialization. Some can be loaded dynamically after SDF is started. To do this, use AOFTREE to define those tree structure entries that will be loaded during initialization, then, use the SDFTREE command to load additional tree structures as needed. For more information, see *IBM Tivoli System Automation for z/OS Programmer's Reference*.

Tree structures loaded after SDF is started must be contained in separate members. Each member must be named after the root component for which the tree structure is defined.

Step 2: Defining SDF Panels

SDF status panels are defined in NetView DSIPARM member AOFPNLS. SA z/OS loads the panel definitions in AOFPNLS when SDF is initialized.

Panel Definition Methods

To define panels in AOFPNLS, you can:

- Use %INCLUDE statements referencing separate NetView DSIPARM members containing panel definitions. This is the recommended method, and the method used in the SA z/OS-provided version of AOFPNLS. See "%INCLUDE Statement for SDF Panels" on page 261 for details on using the %INCLUDE statement for SDF panel definition members.
- Include actual definitions for all panels.
- Use a combination of both %INCLUDE statements and panel definitions.
- Include a subset of panel entries to load during initialization, so that additional panel definitions can be loaded only when needed (see *IBM Tivoli System Automation for z/OS Programmer's Reference*).

System symbols are supported wherever they are used in the AOFTREE, AOFINIT and AOFPNLS members. This can help reduce both customization work and errors.

Panel Definition Structure

The structure of each panel definition is as follows:

- Begin panel definition statement (PANEL)
- Status component definition statements, consisting of pairs of the following statements:
  - STATUSFIELD: defines location of a status component on a panel
  - STATUSTEXT: defines the text displayed in the STATUSFIELD
- Text fields and data definition statements, consisting of pairs of the following statements:
  - TEXTFIELD: defines locations and attributes for constant fields on panels
  - TEXTTEXT: defines text displayed in the TEXTFIELD
- Status panel PF key definitions (PFKnn)
- End panel statement (ENDPANEL)

Descriptions of these panel definition statements are in *IBM Tivoli System Automation for z/OS Programmer's Reference*.

Recommended Order for Defining Panels

When defining panels, it is recommended that you define them in the following order:

1. The root panel
2. The status components for each item listed on the root panel
3. Any other customized status panels

Note: This order of defining panels is a recommendation only. You can define your SDF panels in any order desired.

Example Panel Definition

Figure 45 on page 260 shows how an example SDF panel looks when it is displayed.

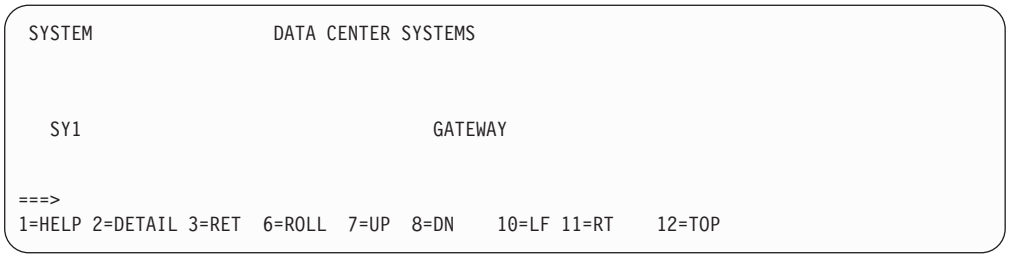


Figure 45. Example SDF Panel

Figure 46 shows the panel definition statements required to define the panel in Figure 45.

```
PANEL(SYSTEM,24,80)
TEXTFIELD(01,02,10,WHITE,NORMAL)
TEXTTEXT(SYSTEM)
TF(01,25,57,WHITE,NORMAL)
TT(DATA CENTER SYSTEMS)
STATUSFIELD(SY1,04,04,11,N,,SY1SYS)
STATUSTEXT(SY1)
SF(SY1.GATEWAY,02,40,47,N,,GATEWAY)
ST(GATEWAY)
TF(24,01,79,T,NORMAL)
TT(1=HELP 2=DETAIL 3=RET  6=ROLL  7=UP 8=DN    ,
10=LF 11=RT 12=TOP)
PFK1(AOCHELP SDF)
PFK2(DETAIL)
PFK3(RETURN)
PFK6(ROLL)
PFK7(UP)
PFK8(DOWN)
PFK10(LEFT)
PFK11(RIGHT)
PFK12(TOP)
ENDPANEL
```

Figure 46. Example Panel Definition Entry

In Figure 46, the panel name is SYSTEM. This panel definition can either be in a separate member referenced by a %INCLUDE statement in AOFPNLS or be directly coded in AOFPNLS. The recommended method is to use a separate member and a %INCLUDE statement. If it is in a separate member, the member name is SYSTEM. You do not have to explicitly define every PF key for the panel. PF key definitions not specified are picked up from definitions in NetView DSIPARM member AOFINIT.

Table 14 describes each statement in Figure 46:

Table 14. Panel Definition Entry Description

Statement	Description and Example Value
`PANEL(SYSTEM,24,80)`	The panel definition statement. The panel name is SYSTEM, the panel length is 24, and the panel width is 80.
`TEXTFIELD(01,02,10,WHITE,NORMAL)`	The text location statement defining constant panel fields. This field starts on line 01 in position 02 and ends in position 10. The color of the field is white and highlighting is normal.
`TEXTTEXT(SYSTEM)`	The text data statement specifying the actual data that goes in the text field just defined. This field contains the word SYSTEM. TEXTFIELD and TEXTTEXT are always grouped in pairs.

Table 14. Panel Definition Entry Description (continued)

Statement	Description and Example Value
TF(01,25,57,WHITE,NORMAL)	Another TEXTFIELD statement for another constant field.
TT(DATA CENTER SYSTEMS)	Another TEXTTEXT statement for the text field just defined.
STATUSFIELD(SY1,04,04,11,N,,SY1SYS)	The location of the status component field. The status component is *SY1*. This field starts on line 04 in position 04 and ends in position 11. The highlighting level is normal. The next panel displayed when the Down PF key is pressed is SY1SYS.
STATUSTEXT(SY1)	The text data used for the name of the field just defined with the STATUSFIELD statement. In this case, the field name is SY1. STATUSFIELD and STATUSTEXT statements are grouped in pairs.
SF(SY1.GATEWAY,02,40,47,N,,GATEWAY)	Another STATUSFIELD definition.
ST(GATEWAY)	Another STATUSTEXT definition.
TF(24,01,79,T,NORMAL) TT(1=HELP 2=DETAIL 3=RET 6=ROLL 7=UP, 8=DN 10=LF 11=RT 12=TOP)	Here, TEXTFIELD and TEXTTEXT are used to display PF key definitions. For this panel, these are the default definitions defined in AOFINIT. If you need values differing from the defaults, there is a statement for defining PF keys unique to this panel, DPFKnn. See *IBM Tivoli System Automation for z/OS Programmer's Reference* for a description of this statement.
PFK1(AOCHELP SDF) PFK2(DETAIL) PFK3(RETURN) PFK6(ROLL) PFK7(UP) PFK8(DOWN) PFK10(LEFT) PFK11(RIGHT) PFK12(TOP)	PF key definition statements.
ENDPANEL	The end panel statement, indicating that this is the end of definitions for this panel.

%INCLUDE Statement for SDF Panels

The %INCLUDE statement for SDF has the following features:

- The SDF %INCLUDE statement allows the specification of a list of members rather than a single member only. Each member name in the list represents a DSIPARM member that is to be loaded. Member names in the list are delimited by a comma.
- The SDF %INCLUDE statement requires parentheses around the specified member or members.
- The target DSIPARM members may contain only complete panel definitions or additional %INCLUDE statements. Panel definitions must be contained within a single member, and therefore cannot be built using commonly defined segments.

System symbols are supported wherever they are used in the AOFTREE, AOFINIT and AOFPNLS members. This can help reduce both customization work and errors.

Step 3: (Optional) Customizing SDF Initialization Parameters

Member AOFINIT allows you to define parameters common to all SDF panels and SDF initialization specifications, such as:

- Initial screen shown when SDF is started
- Maximum operator logon limit
- Default PF key definitions
- Detail status display panel PF key definitions
- Detail status display panel PF key descriptions
- Default priorities and colors

These parameters define values for SDF when it is started.

System symbols are supported wherever they are used in the AOFTREE, AOFINIT and AOFPNLS members. This can help reduce both customization work and errors.

This step of SDF customization is optional. Using SA z/OS-provided default values for these parameters is recommended.

Note: User-defined statuses are not saved across a recycle or a monitor cycle. This means the status of a subsystem will change from the user-defined status to an appropriate SA z/OS status.

Step 4: (Optional) Defining SDF in the Customization Dialog

The SDF entries in the Status Display policy object allow you to define statuses and the priorities assigned to those statuses. These entries are used by SA z/OS common routines to gather data for requests to add status descriptors to status components. The format and values used in SDF Status Detail definitions are described in *IBM Tivoli System Automation for z/OS Programmer's Reference*.

This step of SDF customization is optional. Using SA z/OS-provided definitions for SDF is recommended.

Appendix C. Message Automation

FORCED AT Entry Type

This AT entry must be generated in the predefined way. An AT entry is generated as shown in INGMBASE/INGATCHG (see Figure 47) even though the message may not appear in the Customization Dialog. The AT entry cannot be overridden. No AUTO action is valid.

```
INGMBASE:
*                                    - FORCED AT ENTRY
IF
MSGID = 'EVE172I' & ATF('ING$QRY APPL') ^= ''
 THEN
 EXEC(CMD('EVEEI010 ')ROUTE(ONE %AOFOPGSSOPER%))
 EXEC(CMD('EVEEI009 PPI')ROUTE(ONE %AOFOPGSSOPER%));
*
```

Figure 47. Sample FORCED AT Entry

Because you may need to issue a command or a reply in response to a forced message, you can define a CMD or REPLY for several (but not all) forced messages. This will append an ISSUEACT action to the AT entry, as shown in Figure 48.

```
INGMBASE:
*
* Tape mount monitoring
*                                                      - FORCED AT ENTRY
IF (GROUP:INGTAPE)
MSGID = 'IEF233A'
THEN
EXEC(CMD('INGRTAPE ')ROUTE(ONE %AOFOPSYSOPER%))
DOMACTION(AUTOMATE)
*                                          - CONDITIONAL AT ACTION ENTRY
EXEC(CMD('ISSUEACT ')ROUTE(ONE %AOFOPGSSOPER%));
*
```

Figure 48. Sample FORCED AT Entry with ISSUEACT Action

It is recommended that you refer to INGMBASE/INGATCHG to obtain those AT entries where optional actions are or are not supported.

RECOMMENDED AT Entry Type

You are recommended to use this AT entry in the predefined way. An AT entry is generated as shown in INGMBASE/INGATCHG (see Figure 49 on page 264) even though the message may not have been defined in the Customization Dialog. The AT entry *can* be overridden (using the OVR action) in the Customization Dialog.

```
INGMBASE:
*
* AMRF Buffer recovery
*                                                - RECOMMENDED AT ENTRY
IF
MSGID = 'IEA359E'
THEN
EXEC(CMD('AOFRSA0G ')ROUTE(ONE %AOFOPRECOPER%));
*
* AMRF Buffer recovery
*                                                - RECOMMENDED AT ENTRY
IF
MSGID = 'IEA360A'
THEN
EXEC(CMD('AOFRSA0G ')ROUTE(ONE %AOFOPRECOPER%));
*
* AMRF Buffer recovery
*                                                - RECOMMENDED AT ENTRY
IF
MSGID = 'IEA361I'
THEN
EXEC(CMD('AOFRSA0G ')ROUTE(ONE %AOFOPRECOPER%));
*
```

Figure 49. Sample RECOMMENDED AT Entry Type

Defining a CMD, REP, COD or USR action does not change the recommended AT entry. AUTO(IGNORE) and AUTO(SUPPRESS) prevent an AT entry being created. For other AUT actions the recommended AT entry is built.

CONDITIONAL AT Entry Type

A CONDITIONAL AT entry can be defined for either known or unknown messages.

Known Messages

This AT entry is optional. It is only generated if the message has been defined in the Customization Dialog together with a CMD, REP, AUTO, or OVR action (see Figure 50).

```
INGMBASE:
*
* NETCONV connection is available              (Conditional AT entry)
*
IF
MSGID = 'DUI401I'
 THEN
 EXEC(CMD('ACTIVMSG JOBNAME=NETCONV,UP=YES') ROUTE(ONE %AOFOPGSSOPER%));
*
```

Figure 50. CONDITIONAL AT Entry for a Specific Message

The AT entry is generated as predefined in INGMBASE/INGATCHG for the known message but *can* be overridden with an OVR action. Defining a CMD, REP, or AUTO action does not overrule the predefined behavior (that is, a stop message will still be a stop message, for example).

Unknown Messages

This AT entry is optional. It is only generated if the message has been defined in the Customization Dialog together with a CMD, REP, AUTO, or OVR action.

Any messages that have the same prefix as is checked for in the BEGIN statement, and that do not have a match earlier in the BEGIN-END block, are placed at the end of each block, in front of the trap for outstanding WTORs,

The action statement of the AT entry depends on the action as defined in the Customization Dialog, that is:

Action Statement	Defined Action
ISSUEACT	CMD or REPLY
ACTIVMSG	AUTO(UP, ACTIVE)
TERMMSG	AUTO(ABENDED, BROKEN, TERMINATED, etc.)
HALTMSG	AUTO(HALTED)

This will produce a different AT entry to the standard, specific entry.

Table 15 shows which default AT entry is generated for a particular AUT action.

Table 15. AT Entries That Are Generated by AUT Actions

Status	NetView Automation Table Action Statement
ACTIVE	`EXEC(CMD('ACTIVMSG UP=NO')ROUTE(ONE %AOFOPGSSOPER%))`
ABENDED	`EXEC(CMD('TERMMSG FINAL=YES,ABEND=YES')ROUTE(ONE %AOFOPGSSOPER%))`
ABENDING	`EXEC(CMD('TERMMSG FINAL=NO,ABEND=YES')ROUTE(ONE %AOFOPGSSOPER%))`
BREAKING	`EXEC(CMD('TERMMSG FINAL=NO,BREAK=YES')ROUTE(ONE %AOFOPGSSOPER%))`
BROKEN	`EXEC(CMD('TERMMSG FINAL=YES,BREAK=YES')ROUTE(ONE %AOFOPGSSOPER%))`
CAPTURE	`EXEC(CMD('AOFCPMSG ')ROUTE(ONE %AOFOPGSSOPER%)) DOMACTION(AUTOMATE)`
HALTED	`EXEC(CMD('HALTMSG ')ROUTE(ONE %AOFOPGSSOPER%))`
TERMINATED	`EXEC(CMD('TERMMSG FINAL=YES')ROUTE(ONE %AOFOPGSSOPER%))`
TERMINATING	`EXEC(CMD('TERMMSG FINAL=NO')ROUTE(ONE %AOFOPGSSOPER%))`
UP	`EXEC(CMD('ACTIVMSG UP=YES')ROUTE(ONE %AOFOPGSSOPER%))`

The AT entry can also be defined using the OVR action with the following conditions for its generation:

- If OVR is supported for a predefined AT entry, it will be replaced by the override.
- If OVR is defined multiple times for the same message ID but for different APL instances in the PDB, then multiple AT entries are generated.
- If OVR is defined for a message that other actions have also been defined for, only the OVR AT entry will be generated.
- If OVR is defined for a message at APL CLASS level where *no* check is done for the Jobname (&SUBSJOB), only one OVR AT entry will be generated. (The prerequisite is that at least one instance is linked to that class.)
- If OVR is defined for a message at APL CLASS level where a check for the Jobname (&SUBSJOB) *is* done, one OVR AT entry will be generated for each instance linked to that class.

.

Other Forced AT Entries

The following AT entries are always built:

- BEGIN–END block statements (for performance and design reasons)
- ALWAYS statements
- Capture WTORs

See Figure 51 for examples.

```
INGMBASE:
* ----------------------------------
* Supervisor Messages
IF
MSGID = 'IEA' . & DOMAINID = %AOFDOM%
THEN BEGIN;
*
:
*                                                    - FORCED AT ENTRY
IF
IFRAUWF1(6) = '1'
THEN
EXEC(CMD('OUTREP ')ROUTE(ONE %AOFOPSYSOPER%));
*
ALWAYS
%AOFALWAYSACTION%;
*
END;
* ------------------------------------------------------------------
```

Figure 51. BEGIN–END Block Statements

Restricted Message IDs

The following restricted message IDs will not create an AT entry:

ABCODEPROG	ABCODES	ABCODESYSTM	ABCODETRAN
ABENDED	ABENDING	ACORESTART	ACTCODES
ACTIVE	ALTCODES	AMRFSHORT	AMRFFULL
AMRFCLEAR	AUTODOWN	AUTOTERM	BMPABEND
BREAKING	BRO	BROKEN	CAPMSGS
CHE	CICSINFO	CITIME	CONN
CQSET	CTLDOWN	DATABASE	DOMAINID
DOWN	DN_*xxxxxx*	ENDED	ENDING
EXTSTART	FALLBACK	FORCE	FPABEND
HALFDOWN	HALTED	HEALTHCHK	HOLDQ
IMSINFO	INACTIVE	INGALERT	LISTSHUT
LOGREC	LOGGER	MDSCOUNTA	MDSCOUNTB
MDSCOUNTE	MDSCOUNTF	MDSCOUNTQ	MDSCOUNTR
MDSCOUNTSS	MDSCOUNTSV	MDSCOUNTU	MDSCOUNTV
MDSCOUNTW	MOVED	MVSDUMP	MVSDUMPTAKEN
MVSDUMPRESET	NOJSM	OLDS	OPCA
OPCACMD	OPCAPARM	PPIACTIVE	POSTCHKP
PRECHKP	RCVRTRAN	RECONS	RELEASEQ

RESTART	RESTARTABORT	RUNNING	SHUTFORCEDDF
SHUTTYPES	SMFDUMP	SNAPQ	SPOOLFULL
SPOOLSHORT	STADC	START	STARTED
STARTED2	STOPBMPREGION	STOPFPREGION	STOPPED
STOPPING	STOPREGION	STUCK	SYSLOG
TAPES	TCO	TERMINATING	TERMINATED
TPABEND	UNLKAVM	UNLOCK	UP
UP_*xxxxxx*	USERSTART	VTAMTERMS	VTAMDN
VTAMUP	WORKSTATION	WTOBUF	WTORS
ZOMBIE			

Inheritance Rules for Classes

Bear in mind the following inheritance rules for class data when building AT entries.

Define Application Information

Data is inherited in the APPLICATION INFO policy item per individual field, independent from each other (except for Transient Rerun). If a field is blank, the class value is inherited (if it is available). There are a few exceptions where the inheritance can be blocked without specifying an instance value with the special value NONE , for example, Restart after IPL.

Define Relationships

The External Startup and External Shutdown fields in the sub header area show inherited data individually in the same way as in the APPLICATION INFO policy item. However the relationships are only inherited as a whole if no relationships are defined for the child object.

Define Application Messages and User Data

Data is inherited per message ID. For example, assume a message ID has a command definition for the instance, and the same message ID is defined for a class with reply data. The command and reply data is not merged on the instance, and the class definitions are not inherited at all. Message overrides (OVR) are not inherited at all. All OVRs are used to generate AT entries at the level where they are specified.

Define Startup Procedures

The STARTUP policy offers two panels. The Subsystem Startup Processing panel with a subheader section with input fields that may show inherited values, and for each selected startup phase there is a Startup Command Processing panel with a command input area that also may show inherited data.

Subsystem Startup Processing

Data in the subheader section is inherited per individual field, similar to the APPLICATION INFO policy. Command definitions for the three phases PRESTART, STARTUP, and POSTSTART are inherited per start phase. So if PRESTART commands are defined for the instance and both PRESTART and STARTUP commands are defined for a class, the instance inherits the STARTUP commands from the class.

Startup Command Processing

Within each start phase the commands are inherited all together. So if a PRESTART command is defined for the instance and other PRESTART commands are defined for a class, none of the commands are merged on the instance. Instead the instance has only the one command defined there. No PRESTART commands are inherited from the class.

Define Shutdown Procedures

Shutdown specifications are inherited per phase. So if a SHUTINIT command is defined for the instance and both SHUTINIT and SHUTNORM commands are defined for a class, the instance inherits the SHUTNORM commands from the class. Furthermore, command and reply definitions for one phase are inherited together. So if for SHUTFORCE a command is defined for the instance, and the class has a reply defined for SHUTFORCE, nothing is inherited by that instance.

Changes within inherited data result in creating definitions for the current application. So if for a phase, commands and reply definitions are inherited, and then commands are modified, both the reply and the command definitions become data of the current application. If only commands are inherited for a phase, and then reply data is specified, the command definitions are also copied to the phase definition of the current application.

Define Error Thresholds

The data is inherited as a whole if no thresholds are defined for the child object – it is not possible to specify a level for Critical, Frequent, or Infrequent alone for an instance and inherit the other threshold levels from a class.

Define IMS Subsystem-Specific Data

This policy combines fields that are built into the IMSCNTL and the ENVIRONMENT structures of the ACF. The fields within a structure are inherited all together, but each structure is inherited independently from the other. Furthermore the IMSCNTL fields do not allow definitions for a class (though they are displayed on the class panel). And finally for a subtype other than CTL only a subset of the fields is available.

Thus there are three variations of this panel:

1. Instance of subtype CTL with all IMSCNTL and all ENVIRONMENT fields
2. Class of subtype CTL with all ENVIRONMENT fields
3. Instance or Class of subtype other than CTL with a subset of ENVIRONMENT fields (2 fields)

The first four fields (APPLid, Default HSBID, Startup parm1, Startup parm2) are never inherited. They cannot be specified for a class. The remaining fields are inherited all together in a blocks.

Automatic AT Generation

CMD (Command), REP (Reply), COD (Code), and USR (User Data) are inherited per message ID. For example, assume a message ID has a command definition on the instance, and the same message ID is defined for a class with reply data. The command and reply data are not merged on the instance, and the class definitions are not inherited at all.

Messsage Overrride and Status specifications provide instructions for the generation of the AT entry. This data is never inherited, but is used to create one

AT entry for the object where they are specified. Remember that the AT is message oriented and the AT entry usually has the message ID as a condition, so for example, inheriting a Status would create duplicate entries.

Appendix D. TSO User Monitoring

Active TSO users can be monitored in NMC and SDF using the SA z/OS command DFTSOU (EVJETSOU). To enable TSO user monitoring add the following entry to user AT include fragment INGMSGU1 (or to your own user message table):

```
IF (MSGID='IEF125I' | MSGID='IEF126I' | MSGID='IEF450I')
  THEN EXEC(CMD('DFTSOU UPDATE') ROUTE(ALL *))
  DISPLAY(N) NETLOG(N) CONTINUE(Y);
```

Also, put 'DFTSOU SCAN' in the ACORESTART message for the TSO subsystem.

When DFTSOU is called with the UPDATE parameter then:

- For IEF125I, an ADD request is sent to SDF and NMC for the TSO user that produces the message.
- For IEF126I, a DELETE request is sent to SDF and NMC for the TSO user that produces the message.
- For IEF450I, a DELETE request is sent to SDF and NMC for the failing TSO user. When IEF450I is specified, and the trap is coded in INGMSGU1, then CONTINUE(Y) must also be coded.

When DFTSOU is called with the SCAN parameter, an MVS D TS,L command is issued to identify all currently active TSO users. This data is then passed to SDF and NMC.

NMC updates are associated with NMC object TSO. SDF updates are associated with SDF tree entry TSOUSERS.

Glossary

This glossary includes terms and definitions from:

- The *IBM Dictionary of Computing* New York: McGraw-Hill, 1994.
- The *American National Standard Dictionary for Information Systems* , ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies can be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018. Definitions are identified by the symbol (A) after the definition.
- The *Information Technology Vocabulary* developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.

The following cross-references are used in this glossary:

Contrast with. This refers to a term that has an opposed or substantively different meaning.

Deprecated term for. This indicates that the term should not be used. It refers to a preferred term, which is defined in its proper place in the glossary.

See. This refers the reader to multiple-word terms in which this term appears.

See also. This refers the reader to terms that have a related, but not synonymous, meaning.

Synonym for. This indicates that the term has the same meaning as a preferred term, which is defined in the glossary.

Synonymous with. This is a backward reference from a defined term to all other terms that have the same meaning.

A

ACF. Automation control file.

ACF/NCP. Advanced Communications Function for the Network Control Program. See *Advanced Communications Function* and *Network Control Program*.

ACF/VTAM. Advanced Communications Function for the Virtual Telecommunications Access Method. Synonym for *VTAM*. See *Advanced Communications Function* and *Virtual Telecommunications Access Method*.

active monitoring. In SA z/OS, the acquiring of resource status information by soliciting such information at regular, user-defined intervals. See also *passive monitoring*.

adapter. Hardware card that enables a device, such as a workstation, to communicate with another device, such as a monitor, a printer, or some other I/O device.

Address Space Workflow. In RMF, a measure of how a job uses system resources and the speed at which the job moves through the system. A low workflow indicates that a job has few of the resources it needs and is contending with other jobs for system resources. A high workflow indicates that a job has all the resources it needs to execute.

adjacent hosts. Systems connected in a peer relationship using adjacent NetView sessions for purposes of monitoring and control.

adjacent NetView. In SA z/OS, the system defined as the communication path between two SA z/OS systems that do not have a direct link. An adjacent NetView is used for message forwarding and as a communication link between two SA z/OS systems. For example, the adjacent NetView is used when sending responses from a focal point to a remote system.

Advanced Communications Function (ACF). A group of IBM licensed programs (principally VTAM, TCAM, NCP, and SSP) that use the concepts of Systems Network Architecture (SNA), including distribution of function and resource sharing.

advanced program-to-program communication (APPC). A set of inter-program communication services that support cooperative transaction processing in a Systems Network Architecture (SNA) network. APPC is the implementation, on a given system, of SNA's logical unit type 6.2.

alert. (1) In SNA, a record sent to a system problem management focal point or to a collection point to communicate the existence of an alert condition. (2) In NetView, a high-priority event that warrants immediate

attention. A database record is generated for certain event types that are defined by user-constructed filters.

alert condition. A problem or impending problem for which some or all of the process of problem determination, diagnosis, and resolution is expected to require action at a control point.

alert focal-point system. See entry for NPDA focal-point system under *focal-point system*.

alert threshold. An application or volume service value that determines the level at which SA z/OS changes the associated icon in the graphical interface to the alert color. SA z/OS may also issue an alert. See *warning threshold*.

AMC. (1) Automation Manager Configuration (2) The Auto Msg Classes entry type

APF. Authorized program facility.

API. Application programming interface.

APPC. Advanced program-to-program communications.

application. An z/OS subsystem or job monitored by SA z/OS.

Application entry. A construct, created with the customization dialogs, used to represent and contain policy for an application.

application group. A named set of applications. An application group is part of an SA z/OS enterprise definition and is used for monitoring purposes.

ApplicationGroup entry. A construct, created with the customization dialogs, used to represent and contain policy for an application group.

application program. (1) A program written for or by a user that applies to the user's work, such as a program that does inventory or payroll. (2) A program used to connect and communicate with stations in a network, enabling users to perform application-oriented activities.

ARM. Automatic restart management.

ASCB. Address space control block.

ASCB status. An application status derived by SA z/OS running a routine (the ASCB checker) that searches the z/OS address space control blocks (ASCBs) for address spaces with a particular job name. The job name used by the ASCB checker is the job name defined in the customization dialog for the application.

ASCII (American National Standard Code for Information Interchange). The standard code, using a coded character set consisting of 7-bit coded characters (8-bit including parity check), for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters. (A)

ASF. Automation status file.

authorized program facility (APF). A facility that permits identification of programs that are authorized to use restricted functions.

automated function. SA z/OS automated functions are automation operators, NetView autotasks that are assigned to perform specific automation functions. However, SA z/OS defines its own synonyms, or *automated function names*, for the NetView autotasks, and these function names are referred to in the sample policy databases provided by SA z/OS. For example, the automation operator AUTBASE corresponds to the SA z/OS automated function BASEOPER.

automated console operations (ACO). The concept (versus a product) of using computers to perform a large subset of tasks ordinarily performed by operators, or assisting operators in performing these tasks.

automatic restart management (ARM). A z/OS recovery function that improves the availability of specified subsystems and applications by automatically restarting them under certain circumstances. Automatic restart management is a function of the Cross-System Coupling Facility (XCF) component of z/OS.

automatic restart management element name. In MVS 5.2 or later, z/OS automatic restart management requires the specification of a unique sixteen character name for each address space that registers with it. All automatic restart management policy is defined in terms of the element name, including SA z/OS's interface with it.

automation. The automatic initiation of actions in response to detected conditions or events. SA z/OS provides automation for z/OS applications, z/OS components, and remote systems that run z/OS. SA z/OS also provides tools that can be used to develop additional automation.

automation agent. In SA z/OS, the automation function is split up between the automation manager and the automation agents. The observing, reacting and doing parts are located within the NetView address space, and are known as the ***automation agents***. The automation agents are responsible for:

- recovery processing
- message processing
- active monitoring: they propagate status changes to the automation manager

automation configuration file. The data set that consists of:

- the automation control file (ACF)
- the automation manager configuration file (AMC)
- the NetView automation table (AT)
- the MPFLSTSA member

automation control file (ACF). In SA z/OS, a file that contains system-level automation policy information. There is one master automation control file for each NetView system on which SA z/OS is installed. Additional policy information and all resource status information is contained in the policy database (PDB). The SA z/OS customization dialogs must be used to build the automation control files. They must not be edited manually.

automation flags. In SA z/OS, the automation policy settings that determine the operator functions that are automated for a resource and the times during which automation is active. When SA z/OS is running, automation is controlled by automation flag policy settings and override settings (if any) entered by the operator. Automation flags are set using the customization dialogs.

automation manager. In SA z/OS, the automation function is split up between the automation manager and the automation agents. The coordination, decision making and controlling functions are processed by each sysplex's ***automation manager***.

The automation manager contains a model of all of the automated resources within the sysplex. The automation agents feed the automation manager with status information and perform the actions that the automation manager tells them to.

The automation manager provides ***sysplex-wide*** automation.

Automation Manager Configuration. The Automation Manager Configuration file (AMC) contains an image of the automated systems in a sysplex or of a standalone system.

Automation NetView. In SA z/OS the NetView that performs routine operator tasks with command procedures or uses other ways of automating system and network management, issuing automatic responses to messages and management services units.

automation operator. NetView automation operators are NetView autotasks that are assigned to perform specific automation functions. See also *automated function*. NetView automation operators may receive messages and process automation procedures. There are no logged-on users associated with automation operators. Each automation operator is an operating system task and runs concurrently with other NetView tasks. An automation operator could be set up to handle JES2 messages that schedule automation procedures, and an automation statement could route such messages to the automation operator. Similar to *operator station task*. SA z/OS message monitor tasks and target control tasks are automation operators.

automation policy. The policy information governing automation for individual systems. This includes automation for applications, z/OS subsystems, z/OS data sets, and z/OS components.

automation policy settings. The automation policy information contained in the automation control file. This information is entered using the customization dialogs. You can display or modify these settings using the customization dialogs.

automation procedure. A sequence of commands, packaged as a NetView command list or a command processor written in a high-level language. An automation procedure performs automation functions and runs under NetView.

automation status file. In SA z/OS, a file containing status information for each automated subsystem, component or data set. This information is used by SA z/OS automation when taking action or when determining what action to take. In Release 2 and above of AOC/MVS, status information is also maintained in the operational information base.

automation table (AT). See *NetView automation table*.

autotask. A NetView automation task that receives messages and processes automation procedures. There are no logged-on users associated with autotasks. Each autotask is an operating system task and runs concurrently with other NetView tasks. An autotask could be set up to handle JES2 messages that schedule automation procedures, and an automation statement could route such messages to the autotasks. Similar to *operator station task*. SA z/OS message monitor tasks and target control tasks are autotasks. Also called *automation operator*.

available. In VTAM programs, pertaining to a logical unit that is active, connected, enabled, and not at its session limit.

B

basic mode. A central processor mode that does not use logical partitioning. Contrast with *logically partitioned (LPAR) mode*.

BCP Internal Interface. Processor function of CMOS-390, zSeries processor families. It allows the communication between basic control programs such as z/OS and the processor support element in order to exchange information or to perform processor control functions. Programs using this function can perform hardware operations such as ACTIVATE or SYSTEM RESET.

beaconing. The repeated transmission of a frame or messages (beacon) by a console or workstation upon detection of a line break or outage.

BookManager. An IBM product that lets users view softcopy documents on their workstations.

C

central processor (CP). The part of the computer that contains the sequencing and processing facilities for instruction execution, initial program load (IPL), and other machine operations.

central processor complex (CPC). A physical collection of hardware that consists of central storage, one or more central processors, timers, and channels.

central site. In a distributed data processing network, the central site is usually defined as the focal point for alerts, application design, and remote system management tasks such as problem management.

CFR/CFS and ISC/ISR. I/O operations can display and return data about integrated system channels (ISC) connected to a coupling facility and coupling facility receiver (CFR) channels and coupling facility sender (CFS) channels.

channel. A path along which signals can be sent; for example, data channel, output channel. See also *link*.

channel path identifier. A system-unique value assigned to each channel path.

CHPID. In SA z/OS, channel path ID; the address of a channel.

CHPID port. A label that describes the system name, logical partitions, and channel paths.

channel-attached. (1) Attached directly by I/O channels to a host processor (for example, a channel-attached device). (2) Attached to a controlling unit by cables, rather than by telecommunication lines. Contrast with *link-attached*. Synonymous with *local*.

CI. Console integration.

CICS/VS. Customer Information Control System for Virtual Storage.

CLIST. Command list.

clone. A set of definitions for application instances that are derived from a basic application definition by substituting a number of different system-specific values into the basic definition.

clone ID. A generic means of handling system-specific values such as the MVS SYSCLONE or the VTAM subarea number. Clone IDs can be substituted into application definitions and commands to customize a basic application definition for the system that it is to be instantiated on.

CNC. A channel path that transfers data between a host system image and an ESCON control unit. It can be point-to-point or switchable.

command. A request for the performance of an operation or the execution of a particular program.

command facility. The component of NetView that is a base for command processors that can monitor, control, automate, and improve the operation of a network. The successor to NCCF.

command list (CLIST). (1) A list of commands and statements, written in the NetView command list language or the REXX language, designed to perform a specific function for the user. In its simplest form, a command list is a list of commands. More complex command lists incorporate variable substitution and conditional logic, making the command list more like a conventional program. Command lists are typically interpreted rather than being compiled. (2) In SA z/OS, REXX command lists that can be used for automation procedures.

command procedure. In NetView, either a command list or a command processor.

command processor. A module designed to perform a specific function. Command processors, which can be written in assembler or a high-level language (HLL), are issued as commands.

Command Tree/2. An OS/2-based program that helps you build commands on an OS/2 window, then routes the commands to the destination you specify (such as a 3270 session, a file, a command line, or an application program). It provides the capability for operators to build commands and route them to a specified destination.

common commands. The SA z/OS subset of the CPC operations management commands.

common routine. One of several SA z/OS programs that perform frequently used automation functions. Common routines can be used to create new automation procedures.

Common User Access (CUA) architecture. Guidelines for the dialog between a human and a workstation or terminal.

communication controller. A type of communication control unit whose operations are controlled by one or more programs stored and executed in the unit or by a program executed in a processor to which the controller is connected. It manages the details of line control and the routing of data through a network.

communication line. Deprecated term for *telecommunication line*.

connectivity view. In SA z/OS, a display that uses graphic images for I/O devices and lines to show how they are connected.

console automation. The process of having NetView facilities provide the console input usually handled by the operator.

console connection. In SA z/OS, the 3270 or ASCII (serial) connection between a PS/2 computer and a target system. Through this connection, the workstation appears (to the target system) to be a console.

console integration (CI). A hardware facility that if supported by an operating system, allows operating system messages to be transferred through an internal hardware interface for display on a system console. Conversely, it allows operating system commands entered at a system console to be transferred through an internal hardware interface to the operating system for processing.

consoles. Workstations and 3270-type devices that manage your enterprise.

Control units. Hardware units that control I/O operations for one or more devices. You can view information about control units through I/O operations, and can start or stop data going to them by blocking and unblocking ports.

controller. A unit that controls I/O operations for one or more devices.

couple data set. A data set that is created through the XCF couple data set format utility and, depending on its designated type, is shared by some or all of the z/OS systems in a sysplex. See also *sysplex couple data set* and *XCF couple data set*.

coupling facility. The hardware element that provides high-speed caching, list processing, and locking functions in a sysplex.

CP. Central processor.

CPC. Central processor complex.

CPC operations management commands. A set of commands and responses for controlling the operation of System/390 CPCs.

CPC subset. All or part of a CPC. It contains the minimum *resource* to support a single control program.

CPCB. Command processor control block; an I/O operations internal control block that contains information about the command being processed.

CPU. Central processing unit. Deprecated term for *processor*.

cross-system coupling facility (XCF). XCF is a component of z/OS that provides functions to support cooperation between authorized programs running within a sysplex.

CTC. The channel-to-channel (CTC) channel can communicate with a CTC on another host for intersystem communication.

Customer Information Control System (CICS). A general-purpose transactional program that controls online communication between terminal users and a database for a large number of end users on a real-time basis.

customization dialogs. The customization dialogs are an ISPF application. They are used to customize the enterprise policy, like, for example, the enterprise resources and the relationships between resources, or the automation policy for systems in the enterprise. How to use these dialogs is described in *IBM Tivoli System Automation for z/OS Customizing and Programming*.

CVC. A channel operating in converted (CVC) mode transfers data in blocks and a CBY channel path transfers data in bytes. Converted CVC or CBY channel paths can communicate with a parallel control unit. This resembles a point-to-point parallel path and dedicated connection, regardless whether it passes through a switch.

D

DASD. Direct access storage device.

data services task (DST). The NetView subtask that gathers, records, and manages data in a VSAM file or a network device that contains network management information.

data set. The major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access.

data set members. Members of partitioned data sets that are individually named elements of a larger file that can be retrieved by name.

DBCS. Double-byte character set.

DCCF. Disabled console communication facility.

DCF. Document composition facility.

DELAY Report. An RMF report that shows the activity of each job in the system and the hardware and software resources that are delaying each job.

Devices. You can see information about all devices (such as printers, tape or disk drives, displays, or

communications controllers) attached to a particular switch, and control paths and jobs to devices.

DEVR Report. An RMF report that presents information about the activity of I/O devices that are delaying jobs.

dialog. Interactive 3270 panels.

direct access storage device (DASD). A device in which the access time is effectively independent of the location of the data; for example, a disk.

disabled console communication facility (DCCF). A z/OS component that provides limited-function console communication during system recovery situations.

display. (1) To present information for viewing, usually on the screen of a workstation or on a hardcopy device. (2) Deprecated term for *panel*.

disk operating system (DOS). (1) An operating system for computer systems that use disks and diskettes for auxiliary storage of programs and data. (2) Software for a personal computer that controls the processing of programs. For the IBM Personal Computer, the full name is Personal Computer Disk Operating System (PCDOS).

distribution manager. The component of the NetView program that enables the host system to use, send, and delete files and programs in a network of computers.

domain. (1) An access method and its application programs, communication controllers, connecting lines, modems, and attached workstations. (2) In SNA, a system services control point (SSCP) and the physical units (PUs), logical units (LUs), links, link stations, and associated resources that the SSCP can control by means of activation requests and deactivation requests.

double-byte character set (DBCS). A character set, such as Kanji, in which each character is represented by a 2-byte code.

DP enterprise. Data processing enterprise.

DSIPARM. This file is a collection of members of NetView's customization.

DST. Data Services Task.

E

EBCDIC. Extended binary-coded decimal interchange code. A coded character set consisting of 8-bit coded characters.

ECB. Event control block. A control block used to represent the status of an event.

EMCS. Extended multiple console support.

enterprise. An organization, such as a business or a school, that uses data processing.

enterprise monitoring. Enterprise monitoring is used by SA z/OS to update the *NetView Management Console (NMC)* resource status information that is stored in the *Resource Object Data Manager (RODM)*. Resource status information is acquired by enterprise monitoring of the *Resource Measurement Facility (RMF) Monitor III* service information at user-defined intervals. SA z/OS stores this information in its operational information base, where it is used to update the information presented to the operator in graphic displays.

entries. Resources, such as processors, entered on panels.

entry type. Resources, such as processors or applications, used for automation and monitoring.

environment. Data processing enterprise.

error threshold. An automation policy setting that specifies when SA z/OS should stop trying to restart or recover an application, subsystem or component, or offload a data set.

ESA. Enterprise Systems Architecture.

eServer. Processor family group designator used by the SA z/OS customization dialogs to define a target hardware as member of the zSeries or 390-CMOS processor families.

event. (1) In NetView, a record indicating irregularities of operation in physical elements of a network. (2) An occurrence of significance to a task; for example, the completion of an asynchronous operation, such as an input/output operation. (3) Events are part of a trigger condition, in a way that if all events of a trigger condition have occurred, a *STARTUP* or *SHUTDOWN* of an application is performed.

exception condition. An occurrence on a system that is a deviation from normal operation. SA z/OS monitoring highlights exception conditions and allows an SA z/OS enterprise to be managed by exception.

extended recovery facility (XRF). A facility that minimizes the effect of failures in z/OS, VTAM, the host processor, or high availability applications during sessions between high availability applications and designated terminals. This facility provides an alternate subsystem to take over sessions from the failing subsystem.

F

fallback system. See *secondary system*.

field. A collection of bytes within a record that are logically related and are processed as a unit.

file manager commands. A set of SA z/OS commands that read data from or write data to the automation control file or the operational information base. These commands are useful in the development of automation that uses SA z/OS facilities.

focal point. In NetView, the focal-point domain is the central host domain. It is the central control point for any management services element containing control of the network management data.

focus host. A processor with the role in the context of a unified system image

focal point system. (1) A system that can administer, manage, or control one or more target systems. There are a number of different focal point system associated with IBM automation products. (2) **NMC focal point system**. The NMC focal point system is a NetView system with an attached workstation server and LAN that gathers information about the state of the network. This focal point system uses RODM to store the data it collects in the data model. The information stored in RODM can be accessed from any LAN-connected workstation with NetView Management Console installed. (3) **NPDA focal point system.** This is a NetView system that collects all the NPDA alerts that are generated within your enterprise. It is supported by NetView. If you have SA z/OS installed the NPDA focal point system must be the same as your NMC focal point system. The NPDA focal point system is also known as the *alert focal point system*. (4) **SA z/OS Processor Operations focal point system.** This is a NetView system that has SA z/OS host code installed. The SA z/OS Processor Operations focal point system receives messages from the systems and operator consoles of the machines that it controls. It provides full systems and operations console function for its target systems. It can be used to IPL these systems. Note that some restrictions apply to the Hardware Management Console for an S/390 microprocessor cluster. (5) **SA z/OS SDF focal point system.** The SA z/OS SDF focal point system is an SA z/OS NetView system that collects status information from other SA z/OS NetViews within your enterprise. (6) **Status focal point system.** In NetView, the system to which STATMON, VTAM and NLDM send status information on network resources. If you have a NMC focal point, it must be on the same system as the Status focal point. (7) **Hardware Management Console.** Although not listed as a focal point, the Hardware Management Console acts as a focal point for the console functions of an S/390 microprocessor cluster. Unlike all the other focal points in this definition, the Hardware Management Console runs on a LAN-connected workstation,

frame. For a System/390 microprocessor cluster, a frame contains one or two central processor complexes (CPCs), support elements, and AC power distribution.

full-screen mode. In NetView, a form of panel presentation that makes it possible to display the contents of an entire workstation screen at once. Full-screen mode can be used for fill-in-the-blanks prompting. Contrast with *line mode*.

G

gateway session. An NetView-NetView Task session with another system in which the SA z/OS outbound gateway operator logs onto the other NetView session without human operator intervention. Each end of a gateway session has both an inbound and outbound gateway operator.

generic alert. Encoded alert information that uses code points (defined by IBM and possibly customized by users or application programs) stored at an alert receiver, such as NetView.

generic routines. In SA z/OS, a set of self-contained automation routines that can be called from the NetView automation table, or from user-written automation procedures.

group. A collection of target systems defined through configuration dialogs. An installation might set up a group to refer to a physical site or an organizational or application entity.

group entry. A construct, created with the customization dialogs, used to represent and contain policy for a group.

group entry type. A collection of target systems defined through the customization dialog. An installation might set up a group to refer to a physical site or an organizational entity. Groups can, for example, be of type STANDARD or SYSPLEX.

H

Hardware Management Console. A console used by the operator to monitor and control a System/390 microprocessor cluster.

Hardware Management Console Application (HWMCA). A direct-manipulation object-oriented graphical user interface that provides single point of control and single system image for hardware elements. HWMCA provides customer grouping support, aggregated and real-time system status using colors, consolidated hardware messages support, consolidated operating system messages support, consolidated service support, and hardware commands targeted at a single system, multiple systems, or a customer group of systems.

heartbeat. In SA z/OS, a function that monitors the validity of the status forwarding path between remote systems and the NMC focal point, and monitors the

availability of remote z/OS systems, to ensure that status information displayed on the SA z/OS workstation is current.

help panel. An online panel that tells you how to use a command or another aspect of a product.

hierarchy. In the NetView program, the resource types, display types, and data types that make up the organization, or levels, in a network.

high-level language (HLL). A programming language that does not reflect the structure of any particular computer or operating system. For the NetView program, the high-level languages are PL/I and C.

HLL. High-level language.

host system. In a coupled system or distributed system environment, the system on which the facilities for centralized automation run. SA z/OS publications refer to target systems or focal-point systems instead of hosts.

host (primary processor). The processor at which you enter a command (also known as the *issuing processor*).

HWMCA. Hardware Management Console Application. Application for the graphic hardware management console that monitors and controls a central processor complex. It is attached to a target processor (a system 390 microprocessor cluster) as a dedicated system console. This microprocessor uses OCF to process commands.

I

images. A grouping of processors and I/O devices that you define. You can define a single-image mode that allows a multiprocessor system to function as one central processor image.

IMS/VS. Information Management System/Virtual Storage.

inbound. In SA z/OS, messages sent to the focal-point system from the PC or target system.

inbound gateway operator. The automation operator that receives incoming messages, commands, and responses from the outbound gateway operator at the sending system. The inbound gateway operator handles communications with other systems using a gateway session.

Information Management System/Virtual Storage (IMS/VS). A database/data communication (DB/DC) system that can manage complex databases and networks. Synonymous with IMS.

INGEIO PROC. The I/O operations default procedure name; part of the SYS1.PROCLIB.

initial program load (IPL). (1) The initialization procedure that causes an operating system to commence operation. (2) The process by which a configuration image is loaded into storage at the beginning of a workday or after a system malfunction. (3) The process of loading system programs and preparing a system to run jobs.

initialize automation. SA z/OS-provided automation that issues the correct z/OS start command for each subsystem when SA z/OS is initialized. The automation ensures that subsystems are started in the order specified in the automation control file and that prerequisite applications are functional.

input/output support processor (IOSP). The hardware unit that provides I/O support functions for the primary support processor and maintenance support functions for the processor controller.

Interactive System Productivity Facility (ISPF). An IBM licensed program that serves as a full-screen editor and dialog manager. Used for writing application programs, it provides a means of generating standard screen panels and interactive dialogs between the application programmer and the terminal user.

interested operator list. The list of operators who are to receive messages from a specific target system.

internal token. A *logical token* (LTOK); name by which the I/O resource or object is known; stored in IODF.

IOCDS. I/O configuration data set. The data set that describes the I/O configuration.

I/O Ops. I/O operations.

IOSP. Input/Output Support Processor.

I/O operations. The part of SA z/OS that provides you with a single point of logical control for managing connectivity in your active I/O configurations. I/O operations takes an active role in detecting unusual conditions and lets you view and change paths between a processor and an I/O device, using dynamic switching (the ESCON director). Also known as I/O Ops.

I/O resource number. Combination of channel path identifier (CHPID), device number, etc. See internal token.

IPL. Initial program load.

ISA. Industry Standard Architecture.

ISPF. Interactive System Productivity Facility.

ISPF console. From this 3270-type console you are logged onto ISPF to use the runtime panels for I/O operations and SA z/OS customization panels.

issuing host. See *primary host*; the base program at which you enter a command for processing.

J

JCL. Job control language.

JES. Job entry subsystem.

job. (1) A set of data that completely defines a unit of work for a computer. A job usually includes all necessary computer programs, linkages, files, and instructions to the operating system. (2) An address space.

job control language (JCL). A problem-oriented language designed to express statements in a job that are used to identify the job or describe its requirements to an operating system.

job entry subsystem (JES). A facility for spooling, job queuing, and managing I/O. In SA z/OS publications, JES refers to JES2 or JES3, unless distinguished as being either one or the other.

K

Kanji. An ideographic character set used in Japanese. See also *double-byte character set*.

L

LAN. Local area network.

line mode. A form of screen presentation in which the information is presented a line at a time in the message area of the terminal screen. Contrast with *full-screen mode*.

link. (1) In SNA, the combination of the link connection and the link stations joining network nodes; for example, a System/370 channel and its associated protocols, a serial-by-bit connection under the control of synchronous data link control (SDLC). (2) In SA z/OS, link connection is the physical medium of transmission.

link-attached. Describes devices that are physically connected by a telecommunication line. Contrast with *channel-attached*.

Linux for zSeries and S/390. UNIX-like open source operating system conceived by Linus Torvalds and developed across the internet.

local. Pertaining to a device accessed directly without use of a telecommunication line. Synonymous with *channel-attached*.

local area network (LAN). (1) A network in which a set of devices is connected for communication. They can be connected to a larger network. See also *token ring*. (2) A network in which communications are limited to a moderately-sized geographic area such as a single office building, warehouse, or campus, and that do not generally extend across public rights-of-way.

logical partition (LP). A subset of the processor hardware that is defined to support an operating system. See also *logically partitioned (LPAR) mode*.

logical switch number (LSN). Assigned with the switch parameter of the CHPID macro of the IOCP.

logical token (LTOK). Resource number of an object in the IODF.

logical unit (LU). In SNA, a port through which an end user accesses the SNA network and the functions provided by system services control points (SSCPs). An LU can support at least two sessions — one with an SSCP and one with another LU — and may be capable of supporting many sessions with other LUs. See also *physical unit (PU)* and *system services control point (SSCP)*.

logical unit (LU) 6.2. A type of logical unit that supports general communications between programs in a distributed processing environment. LU 6.2 is characterized by (a) a peer relationship between session partners, (b) efficient use of a session for multiple transactions, (c) comprehensive end-to-end error processing, and (d) a generic application program interface (API) consisting of structured verbs that are mapped into a product implementation. Synonym for advanced program-to-program communications (APPC).

logically partitioned (LPAR) mode. A central processor mode that enables an operator to allocate system processor hardware resources among several logical partitions. Contrast with *basic mode*.

LOGR. The sysplex logger.

LP. Logical partition.

LPAR. Logically partitioned (mode).

LU. Logical unit.

LU-LU session. In SNA, a session between two logical units (LUs) in an SNA network. It provides communication between two end users, or between an end user and an LU services component.

LU 6.2. Logical unit 6.2.

LU 6.2 session. A session initiated by VTAM on behalf of an LU 6.2 application program, or a session initiated by a remote LU in which the application program specifies that VTAM is to control the session by using the APPCCMD macro.

M

MAT. Deprecated term for NetView Automation Table.

MCA. Micro Channel* architecture.

MCS. Multiple console support.

member. A specific function (one or more modules/routines) of a multisystem application that is defined to XCF and assigned to a group by the multisystem application. A member resides on one system in the sysplex and can use XCF services to communicate (send and receive data) with other members of the same group.

message automation table (MAT). Deprecated term for NetView Automation Table.

message class. A number that SA z/OS associates with a message to control routing of the message. During automated operations, the classes associated with each message issued by SA z/OS are compared to the classes assigned to each notification operator. Any operator with a class matching one of the message's classes receives the message.

message forwarding. The SA z/OS process of sending messages generated at an SA z/OS target system to the SA z/OS focal-point system.

message group. Several messages that are displayed together as a unit.

message monitor task. A task that starts and is associated with a number of communications tasks. Message monitor tasks receive inbound messages from a communications task, determine the originating target system, and route the messages to the appropriate target control tasks.

message processing facility (MPF). A z/OS table that screens all messages sent to the z/OS console. The MPF compares these messages with a customer-defined list of messages on which to automate, suppress from the z/OS console display, or both, and marks messages to automate or suppress. Messages are then broadcast on the subsystem interface (SSI).

message suppression. The ability to restrict the amount of message traffic displayed on the z/OS console.

Micro Channel architecture. The rules that define how subsystems and adapters use the Micro Channel bus in a computer. The architecture defines the services that each subsystem can or must provide.

microprocessor. A processor implemented on one or a small number of chips.

migration. Installation of a new version or release of a program to replace an earlier version or release.

MP. Multiprocessor.

MPF. Message processing facility.

MPFLSTSA. The MPFLST member that is built by SA z/OS.

Multiple Virtual Storage (MVS). An IBM licensed program. MVS, which is the predecessor of OS/390, is an operating system that controls the running of programs on a System/390 or System/370 processor. MVS includes an appropriate level of the Data Facility Product (DFP) and Multiple Virtual Storage/Enterprise Systems Architecture System Product Version 5 (MVS/ESA SP5).

multiprocessor (MP). A CPC that can be physically partitioned to form two operating processor complexes.

multisystem application. An application program that has various functions distributed across z/OS images in a multisystem environment.

multisystem environment. An environment in which two or more z/OS images reside in one or more processors, and programs on one image can communication with programs on the other images.

MVS. Multiple Virtual Storage, predecessor of z/OS.

MVS image. A single occurrence of the MVS/ESA operating system that has the ability to process work.

MVS/JES2. Multiple Virtual Storage/Job Entry System 2. A z/OS subsystem that receives jobs into the system, converts them to internal format, selects them for execution, processes their output, and purges them from the system. In an installation with more than one processor, each JES2 processor independently controls its job input, scheduling, and output processing.

MVS/ESA. Multiple Virtual Storage/Enterprise Systems Architecture.

N

NAU. (1) Network accessible unit. (2) Network addressable unit.

NCCF. Network Communications Control Facility.

NCP. (1) Network Control Program (IBM licensed program). Its full name is Advanced Communications Function for the Network Control Program. Synonymous with *ACF/NCP*. (2) Network control program (general term).

NetView. An IBM licensed program used to monitor a network, manage it, and diagnose network problems. NetView consists of a command facility that includes a presentation service, command processors, automation based on command lists, and a transaction processing structure on which the session monitor, hardware

monitor, and terminal access facility (TAF) network management applications are built.

network accessible unit (NAU). A logical unit (LU), physical unit (PU), control point (CP), or system services control point (SSCP). It is the origin or the destination of information transmitted by the path control network. Synonymous with *network addressable unit*.

network addressable unit (NAU). Synonym for *network accessible unit*.

NetView automation procedures. A sequence of commands, packaged as a NetView command list or a command processor written in a high-level language. An automation procedure performs automation functions and runs under the NetView program.

NetView automation table (AT). A table against which the NetView program compares incoming messages. A match with an entry triggers the specified response. SA z/OS entries in the NetView automation table trigger an SA z/OS response to target system conditions. Formerly known as the message automation table (MAT).

NetView Command list language. An interpretive language unique to NetView that is used to write command lists.

NetView (NCCF) console. A 3270-type console for NetView commands and runtime panels for system operations and processor operations.

NetView Graphic Monitor Facility (NGMF). Deprecated term for NetView Management Console.

NetView hardware monitor. The component of NetView that helps identify network problems, such as hardware, software, and microcode, from a central control point using interactive display techniques. Formerly called *network problem determination application*.

NetView log. The log in which NetView records events pertaining to NetView and SA z/OS activities.

NetView message table. See *NetView automation table*.

NetView Management Console (NMC). A function of the NetView program that provides a graphic, topological presentation of a network that is controlled by the NetView program. It provides the operator different views of a network, multiple levels of graphical detail, and dynamic resource status of the network. This function consists of a series of graphic windows that allows you to manage the network interactively. Formerly known as the NetView Graphic Monitor Facility (NGMF).

NetView-NetView task (NNT). The task under which a cross-domain NetView operator session runs. Each NetView program must have a NetView-NetView task to establish one NNT session. See also *operator station task*.

NetView-NetView Task session. A session between two NetView programs that runs under a NetView-NetView Task. In SA z/OS, NetView-NetView Task sessions are used for communication between focal point and remote systems.

NetView paths via logical unit (LU 6.2). A type of network-accessible port (VTAM connection) that enables end users to gain access to SNA network resources and communicate with each other. LU 6.2 permits communication between processor operations and the workstation.

network. (1) An interconnected group of nodes. (2) In data processing, a user application network. See *SNA network*.

Network Communications Control Facility (NCCF). The operations control facility for the network. NCCF consists of a presentation service, command processors, automation based on command lists, and a transaction processing structure on which the network management applications NLDM and NPDA are built. NCCF is a precursor to the NetView command facility.

Network Control Program (NCP). An IBM licensed program that provides communication controller support for single-domain, multiple-domain, and interconnected network capability. Its full name is Advanced Communications Function for the Network Control Program.

Networking NetView. In SA z/OS the NetView that performs network management functions, such as managing the configuration of a network. In SA z/OS it is common to also route alerts to the Networking NetView.

Network Problem Determination Application (NPDA). An NCCF application that helps you identify network problems, such as hardware, software, and microcode, from a central control point using interactive display methods. The alert manager for the network. The precursor of the NetView hardware monitor.

NGMF. Deprecated term for NetView Management Console.

NGMF focal-point system. Deprecated term for NMC focal point system.

NIP. Nucleus initialization program.

NMC focal point system. See *focal point system*

NMC workstation. The NMC workstation is the primary way to dynamically monitor SA z/OS systems. From the windows, you see messages, monitor

status, view trends, and react to changes before they cause problems for end users. You can use multiple windows to monitor multiple views of the system.

NNT. NetView-NetView task.

notification message. An SA z/OS message sent to a human notification operator to provide information about significant automation actions. Notification messages are defined using the customization dialogs.

notification operator. A NetView console operator who is authorized to receive SA z/OS notification messages. Authorization is made through the customization dialogs.

NPDA. Network Problem Determination Application.

NPDA focal-point system. See *focal-point system*.

NTRI. NCP/token-ring interconnection.

nucleus initialization program (NIP). The program that initializes the resident control program; it allows the operator to request last-minute changes to certain options specified during system generation.

O

objective value. An average Workflow or Using value that SA z/OS can calculate for applications from past service data. SA z/OS uses the objective value to calculate warning and alert thresholds when none are explicitly defined.

OCA. In SA z/OS, operator console A, the active operator console for a target system. Contrast with *OCB*.

OCB. In SA z/OS, operator console B, the backup operator console for a target system. Contrast with *OCA*.

OCF. Operations command facility.

OCF-based processor. A central processor complex that uses an operations command facility for interacting with human operators or external programs to perform operations management functions on the CPC.

OPC/A. Operations Planning and Control/Advanced.

OPC/ESA. Operations Planning and Control/Enterprise Systems Architecture.

operating system (OS). Software that controls the execution of programs and that may provide services such as resource allocation, scheduling, input/output control, and data management. Although operating systems are predominantly software, partial hardware implementations are possible. (T)

operations. The real-time control of a hardware device or software function.

operations command facility (OCF). A facility of the central processor complex that accepts and processes operations management commands.

Operations Planning and Control/Advanced (OPC/A). A set of IBM licensed programs that automate, plan, and control batch workload. OPC/A analyzes system and workload status and submits jobs accordingly.

Operations Planning and Control/ESA (OPC/ESA). A set of IBM licensed programs that automate, plan, and control batch workload. OPC/ESA analyzes system and workload status and submits jobs accordingly. The successor to OPC/A.

operator. (1) A person who keeps a system running. (2) A person or program responsible for managing activities controlled by a given piece of software such as z/OS, the NetView program, or IMS. (3) A person who operates a device. (4) In a language statement, the lexical entity that indicates the action to be performed on operands.

operator console. (1) A functional unit containing devices that are used for communications between a computer operator and a computer. (T) (2) A display console used for communication between the operator and the system, used primarily to specify information concerning application programs and I/O operations and to monitor system operation. (3) In SA z/OS, a console that displays output from and sends input to the operating system (z/OS, LINUX, VM, VSE). Also called *operating system console*. In the SA z/OS operator commands and configuration dialogs, OC is used to designate a target system operator console.

operator station task (OST). The NetView task that establishes and maintains the online session with the network operator. There is one operator station task for each network operator who logs on to the NetView program.

operator view. A set of group, system, and resource definitions that are associated together for monitoring purposes. An operator view appears as a graphic display in the graphical interface showing the status of the defined groups, systems, and resources.

OperatorView entry. A construct, created with the customization dialogs, used to represent and contain policy for an operator view.

OS. Operating system.

z/OS component. A part of z/OS that performs a specific z/OS function. In SA z/OS, component refers to entities that are managed by SA z/OS automation.

z/OS subsystem. Software products that augment the z/OS operating system. JES and TSO/E are examples of z/OS subsystems. SA z/OS includes automation for some z/OS subsystems.

z/OS system. A z/OS image together with its associated hardware, which collectively are often referred to simply as a system, or z/OS system.

OSA. I/O operations can display the open system adapter (OSA) channel logical definition, physical attachment, and status. You can configure an OSA channel on or off.

OST. Operator station task.

outbound. In SA z/OS, messages or commands from the focal-point system to the target system.

outbound gateway operator. The automation operator that establishes connections to other systems. The outbound gateway operator handles communications with other systems through a gateway session. The automation operator sends messages, commands, and responses to the inbound gateway operator at the receiving system.

P

page. (1) The portion of a panel that is shown on a display surface at one time. (2) To transfer instructions, data, or both between real storage and external page or auxiliary storage.

panel. (1) A formatted display of information that appears on a terminal screen. Panels are full-screen 3270-type displays with a monospaced font, limited color and graphics. (2) By using SA z/OS panels you can see status, type commands on a command line using a keyboard, configure your system, and passthru to other consoles. See also *help panel*. (3) In computer graphics, a display image that defines the locations and characteristics of display fields on a display surface. Contrast with *screen*.

parallel channels. Parallel channels operate in either byte (BY) or block (BL) mode. You can change connectivity to a parallel channel operating in block mode.

parameter. (1) A variable that is given a constant value for a specified application and that may denote the application. (2) An item in a menu for which the user specifies a value or for which the system provides a value when the menu is interpreted. (3) Data passed to a program or procedure by a user or another program, namely as an operand in a language statement, as an item in a menu, or as a shared data structure.

partition. (1) A fixed-size division of storage. (2) In VSE, a division of the virtual address area that is available for program processing. (3) On an IBM Personal Computer fixed disk, one of four possible storage areas of variable size; one can be accessed by DOS, and each of the others may be assigned to another operating system.

partitionable CPC. A CPC that can be divided into 2 independent CPCs. See also *physical partition, single-image mode, MP, side*.

partitioned data set (PDS). A data set in direct access storage that is divided into partitions, called *members*, each of which can contain a program, part of a program, or data.

passive monitoring. In SA z/OS, the receiving of unsolicited messages from z/OS systems and their resources. These messages can prompt updates to resource status displays. See also *active monitoring.*

PCE. Processor controller. Also known as the “support processor” or “service processor” in some processor families.

PDB. Policy Database

PDS. Partitioned data set.

physical partition. Part of a CPC that operates as a CPC in its own right, with its own copy of the operating system.

physical unit (PU). In SNA, the component that manages and monitors the resources (such as attached links and adjacent link stations) of a node, as requested by a system services control point (SSCP) through an SSCP-PU session. An SSCP activates a session with the physical unit to indirectly manage, through the PU, resources of the node such as attached links.

physically partitioned (PP) configuration. A mode of operation that allows a multiprocessor (MP) system to function as two or more independent CPCs having separate power, water, and maintenance boundaries. Contrast with *single-image (SI) configuration*.

POI. Program operator interface.

policy. The automation and monitoring specifications for an SA z/OS enterprise. See *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

policy database. The database where the automation policy is recorded. Also known as the PDB.

POR. Power-on reset.

port. (1) System hardware to which the I/O devices are attached. (2) On an ESCON switch, a port is an addressable connection. The switch routes data through the ports to the channel or control unit. Each port has a name that can be entered into a switch matrix, and you can use commands to change the switch configuration. (3) An access point (for example, a logical unit) for data entry or exit. (4) A functional unit of a node through

which data can enter or leave a data network. (5) In data communication, that part of a data processor that is dedicated to a single data channel for the purpose of receiving data from or transmitting data to one or more external, remote devices. (6) power-on reset (POR) (7) A function that re-initializes all the hardware in a CPC and loads the internal code that enables the CPC to load and run an operating system.

PP. Physically partitioned (configuration).

PPT. Primary POI task.

primary host. The base program at which you enter a command for processing.

primary POI task (PPT). The NetView subtask that processes all unsolicited messages received from the VTAM program operator interface (POI) and delivers them to the controlling operator or to the command processor. The PPT also processes the initial command specified to execute when NetView is initialized and timer request commands scheduled to execute under the PPT.

primary system. A system is a primary system for an application if the application is normally meant to be running there. SA z/OS starts the application on all the primary systems defined for it.

problem determination. The process of determining the source of a problem; for example, a program component, machine failure, telecommunication facilities, user or contractor-installed programs or equipment, environment failure such as a power loss, or user error.

processor controller. Hardware that provides support and diagnostic functions for the central processors.

processor operations. The part of SA z/OS that monitors and controls processor (hardware) operations. Processor operations provides a connection from a focal-point system to a target system. Through NetView on the focal-point system, processor operations automates operator and system consoles for monitoring and recovering target systems. Also known as ProcOps.

processor operations control file. Named by your system programmer, this file contains configuration and customization information. The programmer records the name of this control file in the processor operations file generation panel ISQDPG01.

Processor Resource/Systems Manager (PR/SM). The feature that allows the processor to use several operating system images simultaneously and provides logical partitioning capability. See also *LPAR*.

ProcOps. Processor operations.

ProcOps Service Machine (PSM). The PSM is a CMS user on a VM host system. It runs a CMS multitasking application that serves as ″virtual hardware″ for ProcOps. ProOps communicates via the PSM with the VM guest systems that are defined as target systems within ProcOps.

product automation. Automation integrated into the base of SA z/OS for the products DB2, CICS, IMS, OPC (formerly called *features*).

program to program interface (PPI). A NetView function that allows user programs to send or receive data buffers from other user programs and to send alerts to the NetView hardware monitor from system and application programs.

protocol. In SNA, the meanings of, and the sequencing rules for, requests and responses used for managing the network, transferring data, and synchronizing the states of network components.

proxy resource. A resource defined like an entry type APL representing a processor operations target system.

PR/SM. Processor Resource/Systems Manager.

PSM. ProcOps Service Machine.

PU. Physical unit.

R

remote system. A system that receives resource status information from an SA z/OS focal-point system. An SA z/OS remote system is defined as part of the same SA z/OS enterprise as the SA z/OS focal-point system to which it is related.

requester. A requester is a workstation software, which enables users to log on to a domain, that is, to the server(s) belonging to this domain, and use the resources in this domain. After the log on to a domain, users can access the shared resources and use the processing capability of the server(s). Because the bigger part of shared resources is on the server(s), users can reduce hardware investment.

resource. (1) Any facility of the computing system or operating system required by a job or task, and including main storage, input/output devices, the processing unit, data sets, and control or processing programs. (2) In NetView, any hardware or software that provides function to the network. (3) In SA z/OS, any z/OS application, z/OS component, job, device, or target system capable of being monitored or automated through SA z/OS.

Resource Access Control Facility (RACF). A program that can provide data security for all your resources. RACF protects data from accidental or deliberate unauthorized disclosure, modification, or destruction.

resource group. A physically partitionable portion of a processor. Also known as a *side*.

Resource Monitoring Facility (RMF) Monitor III. A program that measures and reports on the availability and activity of system hardware and software resources, such as processors, devices, storage, and address spaces. RMF can issue online reports about system performance problems as they occur.

Resource Object Data Manager (RODM). A data cache manager designed to support process control and automation applications. RODM provides an in-memory data cache for maintaining real-time data in an address space that is accessible by multiple applications. RODM also allows an application to query an object and receive a rapid response and act on it.

resource token. A unique internal identifier of an ESCON resource or resource number of the object in the IODF.

restart automation. SA z/OS-provided automation that monitors subsystems to ensure that they are running. If a subsystem fails, SA z/OS attempts to restart it according to the policy in the automation control file.

Restructured Extended Executor (REXX). An interpretive language used to write command lists.

return code. A code returned from a program used to influence the issuing of subsequent instructions.

REXX. Restructured Extended Executor.

REXX procedure. A command list written with the Restructured Extended Executor (REXX), which is an interpretive language.

RMF. Resource Measurement Facility.

RODM. Resource Object Data Manager.

S

SAF. Security Authorization Facility.

SA IOM. System Automation for Integrated Operations Management

SA z/OS. System Automation for z/OS

SA z/OS customization dialogs. An ISPF application through which the SA z/OS policy administrator defines policy for individual z/OS systems and builds automation control data and RODM load function files.

SA z/OS customization focal point system. See *focal point system*.

SA z/OS data model. The set of objects, classes and entity relationships necessary to support the function of SA z/OS and the NetView automation platform.

SA z/OS enterprise. The group of systems and resources defined in the customization dialogs under one enterprise name. An SA z/OS enterprise consists of connected z/OS systems running SA z/OS.

SA z/OS focal point system. See *focal point system*.

SA z/OS policy. The description of the systems and resources that make up an SA z/OS enterprise, together with their monitoring and automation definitions.

SA z/OS policy administrator. The member of the operations staff who is responsible for defining SA z/OS policy.

SA z/OS satellite. If you are running two NetViews on an z/OS system to split the automation and networking functions of NetView, it is common to route alerts to the Networking NetView. For SA z/OS to process alerts properly on the Networking NetView, you must install a subset of SA z/OS code, called an *SA z/OS satellite* on the Networking NetView.

SA z/OS SDF focal point system. See *focal point system*.

SCA. In SA z/OS, system console A, the active system console for a target hardware. Contrast with *SCB*.

SCB. In SA z/OS, system console B, the backup system console for a target hardware. Contrast with *SCA*.

screen. Deprecated term for display panel.

screen handler. In SA z/OS, software that interprets all data to and from a full-screen image of a target system. The interpretation depends on the format of the data on the full-screen image. Every processor and operating system has its own format for the full-screen image. A screen handler controls one PS/2 connection to a target system.

SDF. Status Display Facility.

SDLC. Synchronous data link control.

SDSF. System Display and Search Facility.

secondary system. A system is a secondary system for an application if it is defined to automation on that system, but the application is not normally meant to be running there. Secondary systems are systems to which an application can be moved in the event that one or more of its primary systems are unavailable. SA z/OS does not start the application on its secondary systems.

server. A server is a workstation that shares resources, which include directories, printers, serial devices, and computing powers.

service language command (SLC). The line-oriented command language of processor controllers or service processors.

service processor (SVP). The name given to a processor controller on smaller System/370 processors.

service period. Service periods allow the users to schedule the availability of applications. A service period is a set of time intervals (service windows), during which an application should be active.

service threshold. An SA z/OS policy setting that determines when to notify the operator of deteriorating service for a resource. See also *alert threshold* and *warning threshold*.

session. In SNA, a logical connection between two network addressable units (NAUs) that can be activated, tailored to provide various protocols, and deactivated, as requested. Each session is uniquely identified in a transmission header by a pair of network addresses identifying the origin and destination NAUs of any transmissions exchanged during the session.

session monitor. The component of the NetView program that collects and correlates session-related data and provides online access to this information. The successor to NLDM.

shutdown automation. SA z/OS-provided automation that manages the shutdown process for subsystems by issuing shutdown commands and responding to prompts for additional information.

side. A part of a partitionable CPC that can run as a physical partition and is typically referred to as the A-side or the B-side.

Simple Network Management Protocol (SNMP). An IP based industry standard protocol to monitor and control resources in an IP network.

single image. A processor system capable of being physically partitioned that has not been physically partitioned. Single-image systems can be target hardware processors.

single-image (SI) mode. A mode of operation for a multiprocessor (MP) system that allows it to function as one CPC. By definition, a uniprocessor (UP) operates in single-image mode. Contrast with *physically partitioned (PP) configuration*.

SLC. Service language command.

SMP/E. System Modification Program Extended.

SNA. Systems Network Architecture.

SNA network. In SNA, the part of a user-application network that conforms to the formats and protocols of systems network architecture. It enables reliable transfer of data among end users and provides protocols for controlling the resources of various network configurations. The SNA network consists of network addressable units (NAUs), boundary function components, and the path control network.

SNMP. Simple Network Management Protocol (a TCP/IP protocol). A protocol that allows network management by elements, such as gateways, routers, and hosts. This protocol provides a means of communication between network elements regarding network resources.

solicited message. An SA z/OS message that directly responds to a command. Contrast with *unsolicited message*.

SSCP. System services control point.

SSI. Subsystem interface.

start automation. SA z/OS-provided automation that manages and completes the startup process for subsystems. During this process, SA z/OS replies to prompts for additional information, ensures that the startup process completes within specified time limits, notifies the operator of problems, if necessary, and brings subsystems to an UP (or ready) state.

startup. The point in time at which a subsystem or application is started.

status. The measure of the condition or availability of the resource.

status focal-point system. See *focal-point system*.

status display facility (SDF). The system operations part of SA z/OS that displays status of resources such as applications, gateways, and write-to-operator messages (WTORs) on dynamic color-coded panels. SDF shows spool usage problems and resource data from multiple systems.

steady state automation. The routine monitoring, both for presence and performance, of subsystems, applications, volumes and systems. Steady state automation may respond to messages, performance exceptions and discrepancies between its model of the system and reality.

structure. A construct used by z/OS to map and manage storage on a coupling facility. See cache structure, list structure, and lock structure.

subgroup. A named set of systems. A subgroup is part of an SA z/OS enterprise definition and is used for monitoring purposes.

SubGroup entry. A construct, created with the customization dialogs, used to represent and contain policy for a subgroup.

subplex. Situations where the physical sysplex has been divided into subentities, for example, a test sysplex and a production sysplex. This may be done to isolate the test environment from the production environment.

subsystem. (1) A secondary or subordinate system, usually capable of operating independent of, or asynchronously with, a controlling system. (2) In SA z/OS, an z/OS application or subsystem defined to SA z/OS.

subsystem interface. The z/OS interface over which all messages sent to the z/OS console are broadcast.

support element. A hardware unit that provides communications, monitoring, and diagnostic functions to a central processor complex (CPC).

support processor. Another name given to a processor controller on smaller System/370 processors; see *service processor*.

SVP. Service processor.

switches. ESCON directors are electronic units with ports that dynamically switch to route data to I/O devices. The switches are controlled by I/O operations commands that you enter on a workstation.

switch identifier. The switch device number (swchdevn), the logical switch number (LSN) and the switch name

symbolic destination name (SDN). Used locally at the workstation to relate to the VTAM application name.

synchronous data link control (SDLC). A discipline for managing synchronous, code-transparent, serial-by-bit information transfer over a link connection. Transmission exchanges may be duplex or half-duplex over switched or nonswitched links. The configuration of the link connection may be point-to-point, multipoint, or loop. SDLC conforms to subsets of the Advanced Data Communication Control Procedures (ADCCP) of the American National Standards Institute and High-Level Data Link Control (HDLC) of the International Standards Organization.

SYSINFO Report. An RMF report that presents an overview of the system, its workload, and the total number of jobs using resources or delayed for resources.

SysOps. System operations.

sysplex. A set of z/OS systems communicating and cooperating with each other through certain multisystem hardware components (coupling devices and timers) and software services (couple data sets).

In a sysplex, z/OS provides the coupling services that handle the messages, data, and status for the parts of a multisystem application that has its workload spread across two or more of the connected processors, sysplex timers, coupling facilities, and couple data sets (which contains policy and states for automation).

A Parallel Sysplex is a sysplex that includes a coupling facility.

sysplex application group. A sysplex application group is a grouping of applications that can run on any system in a sysplex.

sysplex couple data set. A couple data set that contains sysplex-wide data about systems, groups, and members that use XCF services. All z/OS systems in a sysplex must have connectivity to the sysplex couple data set. See also *couple data set*.

Sysplex Timer. An IBM unit that synchronizes the time-of-day (TOD) clocks in multiple processors or processor sides. External Time Reference (ETR) is the z/OS generic name for the IBM Sysplex Timer (9037).

system. In SA z/OS, system means a focal point system (z/OS) or a target system (MVS, VM, VSE, LINUX, or CF).

System Automation for Integrated Operations Management. (1) An outboard automation solution for secure remote access to mainframe/distributed systems. Tivoli System Automation for Integrated Operations Management, previously Tivoli AF/REMOTE, allows users to manage mainframe and distributed systems from any location. (2) The full name for SA IOM.

System Automation for OS/390. The full name for SA OS/390, the predecessor to System Automation for z/OS.

System Automation for z/OS. The full name for SA z/OS.

system console. (1) A console, usually having a keyboard and a display screen, that is used by an operator to control and communicate with a system. (2) A logical device used for the operation and control of hardware functions (for example, IPL, alter/display, and reconfiguration). The system console can be assigned to any of the physical displays attached to a processor controller or support processor. (3) In SA z/OS, the hardware system console for processor controllers or service processors of processors connected using SA z/OS. In the SA z/OS operator commands and configuration dialogs, SC is used to designate the system console for a target hardware processor.

System Display and Search Facility (SDSF). An IBM licensed program that provides information about jobs, queues, and printers running under JES2 on a series of panels. Under SA z/OS you can select SDSF from a pull-down menu to see the resources' status, view the z/OS system log, see WTOR messages, and see active jobs on the system.

System entry. A construct, created with the customization dialogs, used to represent and contain policy for a system.

System Modification Program/Extended (SMP/E). An IBM licensed program that facilitates the process of installing and servicing an z/OS system.

system operations. The part of SA z/OS that monitors and controls system operations applications and subsystems such as NetView, SDSF, JES, RMF, TSO, RODM, ACF/VTAM, CICS, IMS, and OPC. Also known as SysOps.

system services control point (SSCP). In SNA, the focal point within an SNA network for managing the configuration, coordinating network operator and problem determination requests, and providing directory support and other session services for end users of the network. Multiple SSCPs, cooperating as peers, can divide the network into domains of control, with each SSCP having a hierarchical control relationship to the physical units and logical units within its domain.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of, networks.

System/390 microprocessor cluster. A configuration that consists of central processor complexes (CPCs) and may have one or more integrated coupling facilities.

T

TAF. Terminal access facility.

target. A processor or system monitored and controlled by a focal-point system.

target control task. In SA z/OS, target control tasks process commands and send data to target systems and workstations through communications tasks. A target control task (a NetView autotask) is assigned to a target system when the target system is initialized.

target hardware. In SA z/OS, the physical hardware on which a target system runs. It can be a single-image or physically partitioned processor. Contrast with *target system*.

target system. (1) In a distributed system environment, a system that is monitored and controlled by the focal-point system. Multiple target systems can be controlled by a single focal-point system. (2) In SA z/OS, a computer system attached to the focal-point system for monitoring and control. The definition of a target system includes how remote sessions are established, what hardware is used, and what operating system is used.

task. (1) A basic unit of work to be accomplished by a computer. (2) In the NetView environment, an operator station task (logged-on operator), automation operator (autotask), application task, or user task. A NetView task performs work in the NetView environment. All SA z/OS tasks are NetView tasks. See also *communications task*, *message monitor task*, and *target control task*.

telecommunication line. Any physical medium, such as a wire or microwave beam, that is used to transmit data.

terminal access facility (TAF). (1) A NetView function that allows you to log onto multiple applications either on your system or other systems. You can define TAF sessions in the SA z/OS customization panels so you don't have to set them up each time you want to use them. (2) In NetView, a facility that allows a network operator to control a number of subsystems. In a full-screen or operator control session, operators can control any combination of subsystems simultaneously.

terminal emulation. The capability of a microcomputer or personal computer to operate as if it were a particular type of terminal linked to a processing unit to access data.

threshold. A value that determines the point at which SA z/OS automation performs a predefined action. See *alert threshold*, *warning threshold*, and *error threshold*.

time of day (TOD). Typically refers to the time-of-day clock.

Time Sharing Option (TSO). An optional configuration of the operating system that provides conversational time sharing from remote stations. It is an interactive service on z/OS, MVS/ESA, and MVS/XA.

Time-Sharing Option/Extended (TSO/E). An option of z/OS that provides conversational timesharing from remote terminals. TSO/E allows a wide variety of users to perform many different kinds of tasks. It can handle short-running applications that use fewer sources as well as long-running applications that require large amounts of resources.

timers. A NetView command that issues a command or command processor (list of commands) at a specified time or time interval.

TOD. Time of day.

token ring. A network with a ring topology that passes tokens from one attaching device to another; for example, the IBM Token-Ring Network product.

TP. Transaction program.

transaction program. In the VTAM program, a program that performs services related to the

processing of a transaction. One or more transaction programs may operate within a VTAM application program that is using the VTAM application program interface (API). In that situation, the transaction program would request services from the applications program using protocols defined by that application program. The application program, in turn, could request services from the VTAM program by issuing the APPCCMD macro instruction.

transitional automation. The actions involved in starting and stopping subsystems and applications that have been defined to SA z/OS. This can include issuing commands and responding to messages.

translating host. Role played by a host that turns a resource number into a token during a unification process.

trigger. Triggers, in combination with events and service periods, are used to control the starting and stopping of applications in a single system or a parallel sysplex.

TSO. Time Sharing Option.

TSO console. From this 3270-type console you are logged onto TSO or ISPF to use the runtime panels for I/O operations and SA z/OS customization panels.

TSO/E. TSO Extensions.

U

UCB. The unit control block; an MVS/ESA data area that represents a device and that is used for allocating devices and controlling I/O operations.

unsolicited message. An SA z/OS message that is not a direct response to a command. Contrast with *solicited message*.

user task. An application of the NetView program defined in a NetView TASK definition statement.

Using. An RMF Monitor III definition. Jobs getting service from hardware resources (processors or devices) are **using** these resources. The use of a resource by an address space can vary from 0% to 100% where 0% indicates no use during a Range period, and 100% indicates that the address space was found using the resource in every sample during that period. See also *Workflow*.

V

view. In the NetView Graphic Monitor Facility, a graphical picture of a network or part of a network. A view consists of nodes connected by links and may also include text and background lines. A view can be displayed, edited, and monitored for status information about network resources.

Virtual Storage Extended (VSE). An IBM licensed program whose full name is Virtual Storage Extended/Advanced Function. It is an operating system that controls the execution of programs.

Virtual Telecommunications Access Method (VTAM). An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability. Its full name is Advanced Communications Function for the Virtual Telecommunications Access Method. Synonymous with *ACF/VTAM*.

VM/ESA. Virtual Machine/Enterprise Systems Architecture.

VM Second Level Systems Support. With this function, Processor Operations is able to control VM second level systems (VM guest systems) in the same way that it controls systems running on real hardware.

volume. A direct access storage device (DASD) volume or a tape volume that serves a system in an SA z/OS enterprise.

volume entry. A construct, created with the customization dialogs, used to represent and contain policy for a volume.

volume group. A named set of volumes. A volume group is part of a system definition and is used for monitoring purposes.

volume group entry. An construct, created with the customization dialogs, used to represent and contain policy for a volume group.

Volume Workflow. The SA z/OS Volume Workflow variable is derived from the RMF Resource Workflow definition, and is used to measure the performance of volumes. SA z/OS calculates Volume Workflow using:

```
                    accumulated
                       Using
Volume     =  ------------------------  * 100
Workflow %    accumulated + accumulated
                Using         Delay
```

The definition of **Using** is the percentage of time when a job has had a request accepted by a channel for the volume, but the request is not yet complete.

The definition of **Delay** is the delay that waiting jobs experience because of contention for the volume. See also *Address Space Workflow*.

VSE. Virtual Storage Extended.

VTAM. Virtual Telecommunications Access Method.

W

warning threshold. An application or volume service value that determines the level at which SA z/OS changes the associated icon in the graphical interface to the warning color. See *alert threshold*.

workflow. See *Address Space Workflow* and *Volume Workflow*.

workstation. In SA z/OS workstation means the *graphic workstation* that an operator uses for day-to-day operations.

write-to-operator (WTO). A request to send a message to an operator at the z/OS operator console. This request is made by an application and is handled by the WTO processor, which is part of the z/OS supervisor program.

write-to-operator-with-reply (WTOR). A request to send a message to an operator at the z/OS operator console that requires a response from the operator. This request is made by an application and is handled by the WTO processor, which is part of the z/OS supervisor program.

WTO. Write-to-Operator.

WTOR. Write-to-Operator-with-Reply.

WWV. The US National Institute of Standards and Technology (NIST) radio station that provides standard time information. A second station, known as WWVB, provides standard time information at a different frequency.

X

XCF. Cross-system coupling facility.

XCF couple data set. The name for the sysplex couple data set prior to MVS/ESA System Product Version 5 Release 1. See also *sysplex couple data set*.

XCF group. A set of related members that a multisystem application defines to XCF. A member is a specific function, or instance, of the application. A member resides on one system and can communicate with other members of the same group across the sysplex.

XRF. Extended recovery facility.

Numerics

390-CMOS. Processor family group designator used in the SA z/OS processor operations documentation and in the online help to identify any of the following S/390 CMOS processor machine types: 9672, 9674, 2003, 3000, or 7060. SA z/OS processor operations uses the OCF facility of these processors to perform operations management functions. See *OCF-based processor*.

Index

Special characters

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Z

Readers' Comments — We'd Like to Hear from You

System Automation for z/OS
Customizing and Programming
Version 3 Release 2

Publication No. SC33-8260-05

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:

- Send your comments to the address on the reverse side of this form.
- Send a fax to the following number: FAX (Germany): 07031+16-3456
  FAX (Other Countries): (+49)+7031-16-3456
- Send your comments via e-mail to: s390id@de.ibm.com

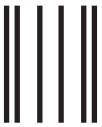
If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	E-mail address

Readers' Comments — We'd Like to Hear from You
SC33-8260-05

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Deutschland Entwicklung GmbH
Department 3248
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Program Number: 5698-SA3

Printed in USA

SC33-8260-05