

System Automation for z/OS
Version 3 Release 4

Programmer's Reference



Note:

Before using this information and the product it supports, read the information in "Notices," on page 341.

This edition applies to IBM Tivoli System Automation for z/OS (5698-SA3) Version 3 Release 4, an IBM licensed program, and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC34-2576-02.

| IBM welcomes your comments. You may forward your comments electronically, or address your comments to:
| IBM Deutschland Research & Development GmbH
| Department 3248
| Schoenaicher Strasse 220
| 71032 Boeblingen
| Germany

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

FAX: (Germany): 07031 16-3456

FAX: (Other countries): +49 7031 16-3456

Internet e-mail: 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.

© Copyright IBM Corporation 1996, 2012.

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

Contents

Figures v

Tables vii

Accessibility ix

Using assistive technologies ix

Keyboard navigation of the user interface ix

z/OS information ix

Dotted decimal syntax diagrams xi

How to send your comments to IBM xiii

If you have a technical problem xiii

About This Book xv

Who Should Use This Book xv

Where to Find More Information xv

Summary of Changes for SC34-2650-00 xvi

Part 1. Introduction 1

Chapter 1. Introduction 3

Overview of Commands 3

Format of Syntax Diagrams 3

**Part 2. SA z/OS System Operations
Commands and Routines 5**

**Chapter 2. SA z/OS System Operations
Commands 7**

Using System Operations Commands for

Programming 7

ACFCMD 7

ACFFQRY 16

ACFREP 21

ACTIVMSG 29

AOCFLT 31

AOCGETCN 33

AOCMSG 34

AOCQRES. 39

AOCQRY 40

AOCUPDT 50

AOFCPMSG 55

AOFEXCMD 58

AOFRACON 60

AOFRCMTR 60

AOFSET 61

AOFTREE 61

CDEMATCH 66

CHKTHRES 69

FWDMSG 72

HALTMSG 74

INGALERT 76

| INGCLEAN 80

INGCNTL 81

INGCPSM 85

INGDATA 87

INGEXEC 91

| INGJLM 95

INGLINK 98

INGMON 101

INGMTRAP 107

INGOMX 108

INGPOST 120

INGQRY 123

INGRCHCK 125

INGRCLUP 126

| INGRDS 126

INGRTCMD 138

INGSIT 139

INGSTOBS 141

INGSTX 144

INGTIMER 148

INGUSS 150

INGVARS 154

INGVSTOP 157

INGVSTRT 158

INGVTAM 159

ISSUEACT (ISSUECMD, ISSUEREP). 161

MDFYSHUT 168

OUTREP 169

TERMMSG 171

Chapter 3. Monitoring Routines 179

AOFADMON 179

AOFAPMON 179

AOFATMON 180

AOFPCPSM 180

AOFNCOMON 181

AOFUXMON 181

INGPJMON 182

INGPSMON 183

| INGROMON 184

INGVMON 185

ISQMTSYS 185

Chapter 4. ING\$QRY NetView

Automation Table Function 187

ING\$QRY 187

**Part 3. SA z/OS I/O Operations
Commands. 191**

**Chapter 5. I/O Operations Commands
(API) 193**

Using I/O Operations Commands for Programming	193
Safe Switching	195
FICON Switches	195
FICON Cascaded Switches	195
Common Elements	196
DELETE FILE	209
QUERY ENTITY CHP	210
QUERY ENTITY CNTLUNIT	215
QUERY ENTITY DEV	218
QUERY ENTITY HOST	222
QUERY ENTITY SWITCH	225
QUERY FILE	228
QUERY INTERFACE CNTLUNIT	229
QUERY INTERFACE SWITCH	235
QUERY RELATION CHP	243
QUERY RELATION CNTLUNIT	244
QUERY RELATION DEV	244
QUERY RELATION HOST	246
QUERY RELATION SWITCH	246
QUERY SWITCH	247
REMOVE and RESTORE CHP	250
REMOVE DEV and RESTORE DEV	254
WRITEFILE	259
WRITEPORT	261
WRITESWCH	266

Chapter 6. Invoking I/O Operations using the API	273
API Calls by REXX EXECs	273
API Calls by the CALL Macro	275

Part 4. Status Display Facility Definitions 281

Chapter 7. SDF Initialization Parameters	283
DCOLOR	283
DPFKnn	284
DPFKDESC1	285
DPFKDESC2	285
EMPTYCOLOR	286
ERRCOLOR	287
INITSCRN	287

MAXOPS	288
PFKnn	288
PRIORITY	290
PRITBLSZ	291
PROPDOWN	292
PROPUP	292
SCREENSZ	292
TEMPERR	293
Priority and Color Default Assignments	294

Chapter 8. SDF Definition Statements 299

AOFTREE	299
BODY	302
BODYHEADER	303
BODYTEXT	305
CELL	305
DATETIME	307
ENDPANEL	308
INPUTFIELD	308
PANEL	309
PFKnn	311
STATUSFIELD	311
STATUSTEXT	314
TEXTFIELD	315
TEXTTEXT	317
Example SDF Definition	318
Example Of A Large SDF Panel	325

Chapter 9. SDF Commands 331

Using SDF Commands	331
SCREEN	333
SDF	333
SDFCONF	334
SDFPANEL	335
SDFTREE	336
Navigation Commands	337

Appendix. Notices 341

Programming Interface Information	342
Trademarks	342

Glossary 345

Index 367

Figures

1.	DISPACF Command Response Panel	12		16.	Example Tree Structure Definitions: System	
2.	Subsystem Dependent Tree	62		SY2		302
3.	Application Tree	64		17.	SDF Example: Tree Structure Definition for	
4.	Code Processing Sample Panel	68		SY1 and SY2		319
	5.	Monitoring a non-SA z/OS controlled job.		98	18.	SDF Example: Hierarchy Defined by SY1 Tree
6.	Sample AT-TLS policy.	118		19.	SDF Example: System Panel Definition	
7.	Exits and Associated Resources	144		Statements		320
8.	INGVARS Command Line-Mode Output	157		20.	SDF Example: Status Component Panel	
9.	INGVTAM REQ=LIST Output	161		Definition Statements for SY1 and SY2		322
10.	INGVTAM subsys REQ=LIST Output	161		21.	Sample SY1 SDF Panel	323
11.	DISPACF Sample Panel	167		22.	SDF definitions of a large display panel	326
12.	Code Processing Panel for an Application			23.	Snapshot of the large display panel (left side)	
Resource	171			followed by Snapshot of the large panel (right		
13.	Code Processing Panel for the MVSESA			side)		328
Resource	171					
14.	QUERY SWITCH Command - Sample Output	250				
	15.	Example Tree Structure Definitions: System				
	SY1	301				

Tables

1.	System Automation for z/OS Library	xv	21.	REMOVE DEV and RESTORE DEV Output	257
2.	Overview of Commands	3	22.	WRITEFILE Input Format	261
3.	Output from ACFFQRY	17	23.	WRITESWCH Input	267
4.	AOCQRY Subsystem Task Global Variables	45	24.	Variables for the PFKnn and DPFKnn Parameters	289
5.	AOCQRY Parent Task Global Variables	46	25.	Table of Default Subsystem Status Colors and Priorities	294
6.	AOCQRY Automation Flag Task Global Variables	48	26.	Table of Default Gateway Status Colors and Priorities	295
7.	Relational Operator Listing	134	27.	Table of Default Spool Status Colors and Priorities	296
8.	TERMMMSG Status Transitions	172	28.	Table of Default WTOR Status Colors and Priorities	296
9.	Standard SA z/OS Array Format	198	29.	Table of Default Monitor Resource Status Colors and Priorities	296
10.	Header for all Query Entity/Interface Output Formats	203	30.	Table of Default Captured Message Status Colors and Priorities	297
11.	Output Format for all Query Relation Commands	204	31.	Table of Default TWS Automation Status Colors and Priorities	297
12.	QUERY ENTITY CHP Output	210	32.	Table of Default Group Status Colors and Priorities	297
13.	QUERY ENTITY CNTLUNIT Output	216	33.	Table of Ensemble Status Colors and Priorities	297
14.	QUERY ENTITY DEV Output	219	34.	Table of Processor Status Colors and Priorities	298
15.	QUERY ENTITY HOST Output	222	35.	Table of LPAR Status Colors and Priorities	298
16.	QUERY ENTITY SWITCH Output	225			
17.	QUERY FILE Output of a Particular Configuration	229			
18.	QUERY INTERFACE CNTLUNIT Output	230			
19.	QUERY INTERFACE SWITCH Output	236			
20.	QUERY SWITCH Output	248			

Accessibility

Publications for this product are offered in Adobe Portable Document Format (PDF) and should be compliant with accessibility standards. If you experience difficulties when using PDF files, you may view the information through the z/OS® Internet Library website or the z/OS Information Center. If you continue to experience problems, send an email to mhvrcfs@us.ibm.com or write to:

IBM® Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
U.S.A.

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 or Library Server versions of z/OS books in the Internet library at:

<http://www.ibm.com/systems/z/os/zos/bkserv/>

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users accessing the Information Center using a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, you know that your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol can be used next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol giving information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, this indicates a reference that is defined elsewhere. The string following the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you should refer to separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:

- ? means an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are

optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

- ! means a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP will be applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.
- * means a syntax element that can be repeated 0 or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3*, 3 HOST, and 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
 2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you could write HOST STATE, but you could not write HOST HOST.
 3. The * symbol is equivalent to a loop-back line in a railroad syntax diagram.
- + means a syntax element that must be included one or more times. A dotted decimal number followed by the + symbol indicates that this syntax element must be included one or more times; that is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can only repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loop-back line in a railroad syntax diagram.

How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or give us any other feedback that you might have.

Use one of the following methods to send us your comments:

1. Send an email to s390id@de.ibm.com
2. Visit the SA z/OS home page at http://www.ibm.com/systems/z/os/zos/features/system_automation/
3. Visit the Contact z/OS web page at <http://www.ibm.com/systems/z/os/zos/webqs.html>
4. Mail the comments to the following address:
IBM Deutschland Research & Development GmbH
Department 3248
Schoenaicher Str. 220
D-71032 Boeblingen
Federal Republic of Germany
5. Fax the comments to us as follows:
From Germany: 07031-16-3456
From all other countries: +(49)-7031-16-3456

Include the following information:

- Your name and address
- Your email address
- Your telephone or fax number
- The publication title and order number:
IBM Tivoli System Automation for z/OS V3R4.0 Programmer's Reference
SC34-2650-00
- The topic and page number related to your comment
- The text of your comment.

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 submit.

If you have a technical problem

Do not use the feedback methods listed above. Instead, do one of the following:

- Contact your IBM service representative
- Call IBM technical support
- Visit the IBM zSeries support web page at www.ibm.com/systems/z/support/.

About This Book

This book describes the programming interfaces of IBM Tivoli® System Automation for z/OS (SA z/OS). It provides detailed reference material you need to operate, maintain and program for SA z/OS.

Throughout this publication references to MVS™ refer either to MVS/ESA, or to the MVS element of z/OS.

Who Should Use This Book

This information is primarily for system programmers and automation programmers, but may also be useful for others, for example, help desk personnel and customer engineers.

Where to Find More Information

The System Automation for z/OS Library

Table 1 shows the information units in the System Automation for z/OS library:

Table 1. System Automation for z/OS Library

Title	Order Number
<i>IBM Tivoli System Automation for z/OS Planning and Installation</i>	SC34-2645
<i>IBM Tivoli System Automation for z/OS Customizing and Programming</i>	SC34-2644
<i>IBM Tivoli System Automation for z/OS Defining Automation Policy</i>	SC34-2646
<i>IBM Tivoli System Automation for z/OS User's Guide</i>	SC34-2647
<i>IBM Tivoli System Automation for z/OS Messages and Codes</i>	SC34-2648
<i>IBM Tivoli System Automation for z/OS Operator's Commands</i>	SC34-2649
<i>IBM Tivoli System Automation for z/OS Programmer's Reference</i>	SC34-2650
<i>IBM Tivoli System Automation for z/OS Product Automation Programmer's Reference and Operator's Guide</i>	SC34-2643
<i>IBM Tivoli System Automation for z/OS TWS Automation Programmer's Reference and Operator's Guide</i>	SC34-2651
<i>IBM Tivoli System Automation for z/OS End-to-End Automation Adapter</i>	SC34-2652
<i>IBM Tivoli System Automation for z/OS Monitoring Agent Configuration and User's Guide</i>	SC34-2653

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/systems/z/os/zos/features/system_automation

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/systems/z/os/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[®], z/VSE[®], and Clusters for AIX[®] and Linux:

- The Internet. You can access IBM message explanations directly from the LookAt Website at www.ibm.com/systems/z/os/zos/bkserv/lookat/index.html
- 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* (SK3T-4271) 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 www.ibm.com/systems/z/os/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* (SK3T-4271).
- The LookAt Website (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 SC34-2650-00

This document contains information previously presented in System Automation for z/OS V3R3.0 Programmer's Reference, SC34-2576-02.

New Information

AOCQRY command

A Subsystem Task Common Global (SUBSIPSTACK) variable and a Parent Task Common Global (SUBPIPSTACK) variable have been added in "AOCQRY" on page 40.

BODYHEADER SDF Definition Statement

BODYHEADER defines a descriptive header and scrolling information for a SDF panel. See "BODYHEADER" on page 303.

BODYTEXT SDF Definition Statement

BODYTEXT defines the text displayed in an SDF panel header. See “BODYTEXT” on page 305.

DATETIME SDF Definition Statement

The DATETIME statement defines the location and attributes of the current date and time. See “DATETIME” on page 307.

DISKonly parameter

The DISKonly parameter is added for the SDFPANEL and SDFTREE command to indicate a panel/tree member is added, and any previously loaded panel/tree member is ignored. See “SDFPANEL” on page 335 or “SDFTREE” on page 336.

INGCLEAN command

The new INGCLEAN command to clean up and repair the in-storage data model. See “INGCLEAN” on page 80.

INGDATA command

The NETVASIS parameter is added for INGDATA. Additional filter criteria are added to the command to match the INGLIST filter criteria. See “INGDATA” on page 87.

INGJLM command

The new INGJLM command starts, stops, suspends, or provides a status on job log monitoring for JES2. See “INGJLM” on page 95.

INGRDS command

The new INGRDS command provides Relational Data Services (RDS) for System Automation. See “INGRDS” on page 126.

INGROMON monitoring routine

The new INGROMON monitoring routine is added. See “INGROMON” on page 184.

SDF command

The SDF command to launch the SDF component is added for the SDF commands. See “SDF” on page 333.

SDFCONF command

The SDFCONF command to delete an SDF record with a confirmation screen is added. See “SDFCONF” on page 334.

SDF Navigation Commands

“Navigation Commands” on page 337 within an SDF panel hierarchy are added in Chapter 9, “SDF Commands,” on page 331.

Changed Information

AOFCPMSG command

The message actions for the NORMAL value of the SEVERITY parameter are amended. The DOM parameter is updated to include a facility to overwrite or delete messages, as they become obsolete. See “AOFCPMSG” on page 55.

CDEMATCH command

The parameters, return codes, usage notes, task global variables, and example. The VARn parameter is added. See “CDEMATCH” on page 66.

INGALERT command

The USRn and CDEn parameters are added. See “INGALERT” on page 76.

INGQRY command

A new attribute IPSTACK is added. See “INGQRY” on page 123.

ING\$QRY command

A new attribute IPSTACK is added. The *attribute* parameter has several new values. See Chapter 4, “ING\$QRY NetView Automation Table Function,” on page 187.

INGCNTL command

The examples have been expanded and updated. The ALERTHOST and TTHOST parameters are modified. The EIFPPI parameter is updated. See “INGCNTL” on page 81.

INGDATA command

The CATEGORY and SUBTYPE parameters are added. See “INGDATA” on page 87.

INGEXEC command

The CORRWAIT, DESCR, NETVASIS and TERMMSG parameters are added to “INGEXEC” on page 91.

INGLINK command

The usage of the *user_data* parameter has changed. Use of the new ISUP_ and ISDN_ messages for application linking is described and screen examples are updated to show additional column displays for the status of these messages. See “INGLINK” on page 98.

INGMON command

The INACTIVE status for Monitor Status is removed and status transitions are redefined. See “INGMON” on page 101.

INGOMX command

An additional example is provided for a Secure Socket Connection. Use of INGOMX directives is added for handling SOAP requests. See “INGOMX” on page 108.

INGRCLUP command

The *jobname* parameter takes wildcards and the restrictions have been updated. See “INGRCLUP” on page 126.

INGUSS command

The STDENV parameter is added for assigning a path for environment variables. The NETVASIS parameter is added for entering commands in mixed or lower case. Return codes are added. See “INGUSS” on page 150.

MDFYSHUT command

The ABORT parameter is added to the MDFYSHUT command. See “MDFYSHUT” on page 168.

QUERY SWITCH

The name LAIBLEDTEXT is added to the “QUERY SWITCH” on page 247 output.

SCREENSZ initialization parameter

The “SCREENSZ” on page 292 initialization parameter is amended to include new screen and panel size definition capacity.

Example Of A Large SDF Panel for SDF definition statements

A further example is added to Chapter 8, “SDF Definition Statements,” on page 299 to demonstrate enhanced screen and panel display definitions. See “Example Of A Large SDF Panel” on page 325.

Moved Information

Priority and Color Default Assignments

The priority and color assignment tables for subsystems are taken from the *IBM Tivoli System Automation for z/OS Defining Automation Policy* and added here. An additional table for ProcOps color assignment within SDF is added. Refer to “Priority and Color Default Assignments” on page 294.

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.

Part 1. Introduction

Chapter 1. Introduction	3	Format of Syntax Diagrams	3
Overview of Commands	3		

This part describes System Automation for z/OS commands in general—how to enter them, the format, and the various types of commands, and also explains the format of syntax diagrams that are used for commands.

Chapter 1. Introduction

Overview of Commands

Table 2 gives a brief overview of the System Automation for z/OS commands. This overview lists the various types of commands, their functions and where they can be entered.

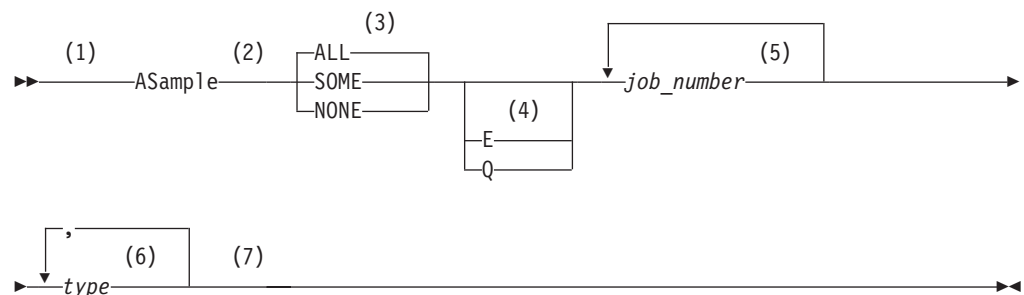
Table 2. Overview of Commands

Type of command	Function	Where entered
System operations commands	Control and maintain resources in the enterprise from a single point of control	NetView console, or NMC
I/O operations commands	Control input/output devices	TSO/ISPF, API, operator console
Processor operations commands	Common commands for automation	API, NetView console, or NMC Note: Precede with ISQCCMD command
Control hardware processors	NetView console or NMC	

Format of Syntax Diagrams

The description of each command and routine includes the format of the command in a syntax diagram. The diagram shows the operands for the commands. Use blanks to separate the operands, unless otherwise stated or diagrammed.

To construct a command from the diagram, follow the diagram from left to right, choosing the path that suits your needs. The following is a sample syntax diagram with notes that explain how to use it to construct a command. This command is for illustration only. Do not attempt to enter it.



Notes:

- 1 Start here. ►► indicates the start of the diagram.
- 2 Type ASAMPLE, or abbreviate it to AS. The uppercase characters are the abbreviation. Operands on the main line are required.
- 3 Choose one of the options. The default is always above the main line. In this case, ALL is the default. If the option includes punctuation marks, include them too, for example: = () . ,

- 4 Choose E, Q, or neither. Operands below the main line are optional.
- 5 Repeat *job_number* any number of times. Variables are shown in italics. Replace them with a real name or value.
- 6 Repeat *type* any number of times, separated with a comma. Variables are shown in italics. Replace them with a real name or value.
- 7 End here. →◀ indicates the end of the command.

If a command continues to the next line, you see → and ▶.

|and| indicates a fragment for a specific condition or option.

The following examples show commands that you might enter, based on the sample syntax diagram:

```
asample none q DAF00821 DAF00832 ELD00824
```

```
as some DLR01445
```


Part 2. SA z/OS System Operations Commands and Routines

Chapter 2. SA z/OS System Operations	
Commands	7
Using System Operations Commands for Programming	7
ACFCMD	7
ACFFQRY	16
ACFREP	21
ACTIVMSG	29
AOCFILT	31
AOCGETCN	33
AOCMSG	34
AOCQRES.	39
AOCQRY	40
AOCUPDT	50
AOFCPMSG	55
AOFEXCMD	58
AOFRACON	60
AOFRCMTR	60
AOFSET	61
AOFTREE	61
CDEMATCH	66
CHKTHRES	69
FWDMMSG	72
HALTMSG	74
INGALERT	76
INGCLEAN	80
INGCNTL	81
INGCPSM	85
INGDATA	87
INGEXEC	91
INGJLM	95
INGLINK	98
INGMON	101
INGMTRAP	107
INGOMX	108
INGPOST	120
INGQRY	123
INGRCHCK	125
INGRCLUP	126
INGRDS	126
INGRTCMD	138
INGSIT	139
INGSTOBS	141
INGSTX	144
INGTIMER	148
INGUSS	150
INGVARS	154
INGVSTOP	157
INGVSTRT	158
INGVTAM	159
ISSUEACT (ISSUECMD, ISSUEREP)	161
MDFYSHUT	168
OUTREP	169
TERMMSG	171
Chapter 3. Monitoring Routines	179
AOFADMON	179
AOFAPMON	179
AOFATMON	180
AOFCPSPM	180
AOFNCMON	181
AOFUXMON	181
INGPJMON	182
INGPSMON	183
INGROMON	184
INGVMON	185
ISQMTSYS	185
Chapter 4. ING\$QRY NetView Automation Table Function	187
ING\$QRY	187

This part describes System Automation for z/OS commands and routines, including specifics of how to enter them and their format.

See *IBM Tivoli System Automation for z/OS User's Guide* for general information about SA z/OS commands.

Chapter 2. SA z/OS System Operations Commands

Using System Operations Commands for Programming

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

These commands are provided either as building blocks that you must incorporate into your script, or as complete routines that you can call from Network Communications Control Facility (NCCF), the NetView automation table (AT), from timers, or from other automation procedures without further programming. If a command can be used as a complete routine, this is specified in the usage notes.

Using these commands in automation procedures provides you with the following advantages:

- Reduced development time: Less code has to be written.
- Portable code: Automation policy information that is unique to an enterprise can be kept in the automation control file rather than distributed among many automation procedures. The automation procedures implement a number of different rules for handling a situation and the automation control file is used to select which rules are applicable to the current situation.
- A consistent, documented interface.

The commands that are described here may be used while automating any SA z/OS application. In the context of SA z/OS, an *application* is defined as:

- An MVS subsystem
- An MVS job
- A non-MVS resource, that is, a resource that is not a z/OS address space, or that does not respond to the usual MVS startup and shutdown commands
- Your own applications

Occasionally, you may see the term *subsystem* used to refer to applications in general.

ACFCMD

Purpose

The ACFCMD command allows an automation procedure to issue commands defined in the automation policy. It searches the automation control file for the specified entries, performs variable substitution for predefined variables, then issues the commands.

ACFCMD can also issue commands that are built dynamically by the calling automation procedure and passed to ACFCMD through a special task global variable named EHKCMD.

ACFCMD

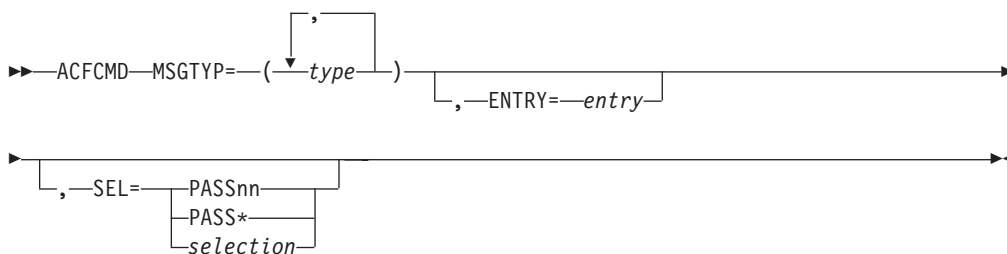
In general you should consider using ISSUECMD or ISSUEACT from the automation table, rather than calling ACFCMD directly. This has the following advantages:

- It checks the automation flags for you, to ensure that automation is allowed.
- It checks that the job that issued the message is known to SA z/OS.

Syntax

To issue commands that are directly defined in the automation control file use the following syntax:

1. Syntax for directly defined commands



To issue commands built dynamically by the calling automation procedure use the following syntax:

2. Syntax for dynamically built commands



Parameters

MSGTYP=*type*

This value is the message ID in the MESSAGES/USER DATA policy item where the commands to be issued by ACFCMD are defined. The value of MSGTYP is typically coded with the message ID or with a generic name such as SPOOLSHORT or SPOOLFULL. The specified *type* values are searched in the order specified until an entry with the given entry and type value can be found.

ENTRY=*entry*

This value is the entry in the automation MESSAGES/USER DATA policy item where the commands to be issued are defined. The default is the subsystem name, if the commands are issued for applications.

This parameter is mutually exclusive with the FUNC=ISSUE parameter.

SEL=

This parameter provides the criteria for the first field in the command entry. This field gives detailed criteria to select a command or commands from the automation control file. Based on the MSGTYP, ENTRY and SEL fields, any specific command can be retrieved from a group of commands associated with a message entry. This parameter is mutually exclusive with the FUNC=ISSUE parameter.

The commands associated with the specific pass selection value defined in the automation policy are issued, along with all commands defined without a selection value. For selection values beginning with PASS, additionally those commands with the pass selection value of PASS* are issued.

IF no SEL parameter is coded, all commands are selected without respect to any pass selection value in the first field of the command entry.

PASS nn

PASS nn values can range from 1 through 99 and must be coded without leading zeros, such as PASS1, PASS2, and PASS3.

When SEL=PASS nn is specified, commands associated with the PASS nn selection value defined in the automation policy are issued, along with all commands defined with a selection value of PASS* or with no selection value.

PASS*

When SEL=PASS* is specified, commands associated with the PASS* selection value defined in the automation policy are issued, along with all commands defined with a selection value beginning with the prefix PASS or with no selection value.

selection

When SEL=*selection* is specified, the commands associated with the specific selection value defined in the automation policy are issued, along with all commands defined without a selection value.

FUNC=ISSUE

The command to be issued is taken from the task global variable EHKCMD.

This parameter is mutually exclusive with the ENTRY and SEL parameters.

Restrictions and Limitations

The ACFCMD command should be called by an automation procedure or by a command processor. The AOCQRY command must be invoked first to set the SUBSAPPL and SUBSTYPE task global variables.

The ACF COLD command temporarily disables automation. ACFCMD will not work while the automation control file is being reloaded. This is necessary to ensure that the SA z/OS environment, as defined by the reloaded automation control file, is established correctly. Full automation resumes when the A0F540I - INITIALIZATION RELATED PROCESSING HAS BEEN COMPLETED message has been received.

Return Codes

- 0 At least one command was found and issued.
- 1 No commands meeting the selection criteria were found.
- 2 The issued command returned a non-zero return code and return code checking was enabled through the customization dialogs.
- 4 Invalid parameters were used in the call.
- 6 SA z/OS initialization incomplete, unable to process command request.

Usage

- If the command that is issued uses symbols you should call AOCQRY to substitute or translate the symbols. Refer to the tables in "Task Global Variables" on page 44 for AOCQRY.

- ACFCMD can issue multiple commands during a single instance of processing. Commands can be defined to one ENTRY and MSGTYP combination but with different or duplicate selection fields. During processing all selection fields are located that match the selection criteria and their associated commands are issued in the same sequence that they have been defined in the automation policy.
- When FUNC=ISSUE is used the ACFCMD command can issue only one command during a single instance of processing.
- SA z/OS variable CMDCNTHI is returned to the calling automation procedure as a task global variable value. ACFCMD retrieves all command entries for a given ENTRY/MSGTYP and searches for the highest PASS nn number. The highest PASS nn number is returned in CMDCNTHI. You can use this number to determine whether all available commands are issued and an appropriate error message should be issued to the operator. If PASS nn is not coded, CMDCNTHI is zero.
- Variables are available to change the command entered in the automation control file. Variables &EHKVAR0 through &EHKVAR9 and &EHKVART must be defined as task global variables in the calling automation procedure and must be initialized with the data to change the commands. These variables are passed to ACFCMD. Whenever ACFCMD finds a detail command entry in the automation control file it scans the command entry looking for &EHKVAR n . If an &EHKVAR n variable is found, the value stored in the automation procedure variable replaces the &EHKVAR n in the command entry. Multiple &EHKVAR n variables can be coded in a single command entry. Delimiters are unnecessary, and the variables can be coded between any other text.

Task Global Variables

The following run time variables are used by ACFCMD for different purposes:

CMDCNTHI

This variable contains the number of the highest PASS nn selection for defined commands to the specified entry and type in the automation policy.

EHKCMD

When ACFCMD is called with FUNC=ISSUE, this variable must provide the command that is to be issued.

EHKCMDTEXT

Text for confirmation message AOF570I, after having issued the command.

EHKVAR0 through EHKVAR9 and EHKVART

These variables can be used to alter command definitions in the automation policy. If included in the defined commands with a leading &, the variables are substituted by their values before the commands are issued. The values of these variables must be provided by the calling automation routine.

&APPLPARMS

This variable provides the value that is entered in the APPLPARMS parameter of the INGREQ command when starting or stopping the resource related to the specified entry. The value of this variable is only available during startup or shutdown processing of a resource under the control of SA z/OS and can be used to alter commands entered in the automation policy.

If the AOCQRY command has been invoked in the calling automation routine, it sets the following task global variables if the appropriate information is applicable.

These variables represent only a subset of those that can be used to alter commands that are entered in the automation policy (where *x* can be either S for subsystem, or P for parent subsystem):

- SUB x APPL
- SUB x ASID
- SUB x CATEGORY
- SUB x CMDPFX
- SUB x DESC
- SUB x FILE
- SUB x FILTER
- SUB x IPSTACK
- SUB x JOB
- SUB x PATH
- SUB x PID
- SUB x PLEX
- SUB x PORT
- SUB x PROC
- SUB x SCHEDSS
- SUB x SHUTDLY
- SUB x SPARM
- SUB x SUBCAT
- SUB x SUBID
- SUB x SUBTYPE
- SUB x SYMBOL n ($n=1-9$)
- SUB x USER
- SUB x USSJOB

These task global variables are substituted when they are specified with a leading & in commands that have been defined in the automation policy.

Valid symbols for provider run time variables are as follows (where *x* can be either S for subsystem, or P for parent subsystem):

- SU2 x APPL
- SU2 x ASID
- SU2 x CATEGORY
- SU2 x CMDPFX
- SU2 x DESC
- SU2 x FILE
- SU2 x FILTER
- SU2 x IPSTACK
- SU2 x JOB
- SU2 x PATH
- SU2 x PID
- SU2 x PORT
- SU2 x PROC
- SU2 x SCHEDSS
- SU2 x SHUTDLY
- SU2 x SUBCAT
- SU2 x SUBID
- SU2 x SUBTYPE
- SU2 x SYMBOL n ($n=1-9$)
- SU2 x USER
- SU2 x USSJOB

Common Global Variables

Common global variable AOFJESPREFIX is substituted in the command to be issued when found.

Examples

Example 1

This example shows the relationship between ACFCMD and the automation control file. The message to automate, \$HASP607, is produced by the JES2 subsystem and indicates that JES2 is not dormant. The automation procedure responds to this by calling ACFCMD to issue a command to stop the JES2 initiators, (MVS \$PI).

The command is defined in the automation policy through the customization dialog panels.

If you enter DISPACF JES2 \$HASP607 a panel with information similar to Figure 1 is displayed.

```
Command = ACF ENTRY=JES2,TYPE=$HASP607,REQ=DISP
SYSTEM = KEY3      AUTOMATION CONFIGURATION DISPLAY - ENTRY= JES2
-----
AUTOMATION CONFIGURATION DISPLAY - ENTRY= JES2
TYPE IS $HASP607
CMD      = (,,'MVS $PI')
END OF MULTI-LINE MESSAGE GROUP
```

Figure 1. DISPACF Command Response Panel

The automation procedure to issue this command is:

```
/* REXX CLIST to automate $HASP607 */
/* Check whether automation allowed and set TGLOBALs */
'AOCQRY ...'
:
'ACFCMD MSGTYP=$HASP607,ENTRY=JES2'
Select
  When rc = 0 Then Nop /* Command issued OK */
  When rc = 1 Then Do /* No commands issued; warn if required */
:
  End
  Otherwise Do /* Error; perform warning action */
:
  End
End
Exit
```

ACFCMD uses the parameters passed to it to find the corresponding values in the automation policy. Because no SEL parameter is coded, no selection restriction is made with respect to the first field of the command entry.

Upon return to the automation procedure, the rc special variable is checked to ensure a command was found in the automation control file. The automation procedure takes appropriate action if a command is not found or a processing error occurs in the ACFCMD command.

Example 2

This example uses the same scenario as Example 1, but shows how you can use defaults to minimize coding. The message to automate, \$HASP607, is produced by

the JES2 subsystem and indicates that JES2 is not dormant. The automation procedure responds by calling ACFCMD to issue a command to stop the JES2 initiators (\$PI).

The command is defined in the automation policy as in Example 1.

The automation procedure to issue this command is:

```

/* REXX CLIST to automate $HASP607                                */
/* Check whether automation allowed and set TGLOBALs             */
'AOCQRY ...'                                                    */
:
'ACFCMD MSGTYP='Msgid()
Select
  When rc = 0 Then Nop      /* Command issued OK                */
  When rc = 1 Then Do      /* No commands issued; warn if required */
:
  End
  Otherwise Do              /* Error; perform warning action      */
:
  End
End
Exit

```

This example differs from Example 1 in the following ways:

- ACFCMD uses a NetView REXX function for the MSGTYP field, assumes defaults for the ENTRY and SEL fields and uses task global variables set up by AOCQRY for the ENTRY default.
- The ENTRY field defaults to JES2 because the job name on the message was the job name for the JES2 subsystem, so the SUBSAPPL task global (which is the default entry type) currently contains JES2. The AOCQRY command must be called before ACFCMD for the ENTRY default to work correctly.
- The MSGTYP field uses the NetView REXX function Msgid(), which contains the message identifier for the message that called the automation procedure. This message identifier is supplied only to an automation procedure called from the NetView automation table. This value can be used when calling ACFCMD.

Note: If your code issues a WAIT command before it issues the ACFCMD you must store the msgid() value in a temporary global as the NetView MSGREAD command overwrites the data from the message that invoked the procedure.

Assuming that AOCQRY is invoked to check the Terminate flag, both of the above examples are equivalent to invoking the following from the NetView automation table for \$HASP607:

```
ISSUECMD AUTOTYP=TERMINATE
```

Example 3

This example shows the use of *PASS_m* logic in an automation procedure. The message to automate, \$HASP607, is produced by the JES2 subsystem and indicates that JES2 is not dormant. The automation procedure responds the first time by stopping the JES2 initiators (\$PI command), and the second time by abending JES2 (\$P JES2,ABEND).

The commands are defined in the automation policy through the customization dialogs. The data is stored in the automation control file in the following way:

```

AUTOMATION CONFIGURATION DISPLAY - ENTRY= JES2
TYPE IS $HASP607
CMD          = (PASS1,, 'MVS $PI')
CMD          = (PASS2,, 'MVS $P JES2, ABEND')
END OF MULTI-LINE MESSAGE GROUP

```

The automation procedure to issue the commands is:

```

/* REXX CLIST to automate $HASP607 */
/* Check whether automation allowed and set TGLOBALs */
'AOCQRY ...'
:
/* Increase the counter unique to this automation procedure */
'GLOBALV GETC HASP607_CNT'
If hasp607_cnt = '' Then hasp607_cnt = 1
Else hasp607_cnt = hasp607_cnt + 1
'GLOBALV PUTC HASP607_CNT'
/* Issue the ACF command for the pass number as determined */
'ACFCMD MSGTYP='Msgid()',SEL=PASS'hasp607_cnt
Select
  When rc = 0 Then Nop /* Command issued OK */
  When rc = 1 Then Do /* No commands issued; warn if required */
  :
  End
  Otherwise Do /* Error; perform warning action */
  :
  End
End
Exit

```

This example differs from the previous examples in the following ways:

- The automation procedure uses a unique common global variable, in this case HASP607_CNT, to maintain a PASS counter. The automation procedure adds 1 to this counter each time it is processed, then appends the counter to the SEL=PASS field. During processing, the counter is translated, and PASS1 or PASS2 is processed. Note that a null test is required to set the counter to 1 if it has not been set before. If the counter exceeds 2 then the ACFCMD will set a return code of 1 since there is no matching entry in the automation control file.

Note: This example assumes you are using one JES subsystem. If you are using multiple JES subsystems, you must use a different counter variable for each.

- Another automation procedure that resets the counter is necessary to complete the logic flow. For this example, the automation procedure runs when the final JES2 message or a startup message is received. Note that the counter is cleared rather than set to zero. This saves an entry in the NetView global dictionary unless the message \$HASP607 has occurred.

The automation procedure to reset the counter is:

```

/* REXX CLIST to reset the counter */
hasp607_cnt = ''
'GLOBALV PUTC HASP607_CNT'
Exit

```

Notes:

1. To ensure serialization of access to the NetView global dictionary and the correct ordering of the commands issued, the NetView automation table entry should route the command to a specific operator if the message may occur more than once in quick succession.
2. If AOCQRY is checking the Terminate flag this example could be coded as:

```
ISSUECMD AUTOTYP=TERMINATE,PASSES=YES
```

The pass count will be reset when the application final termination message is processed.

Example 4

This example shows the use of `EHKVAR n` variables. It also shows the use of duplicate selection fields because two entries are coded, each with `PASS1`. The message to automate is given in response to the `JES2 $DU` command, which displays all `JES2` devices. The message ID produced by `JES2` is `$HASP628`. The example assumes the full text of the message is passed to the automation procedure. The automation procedure checks the resource type, and if the resource is a line, stops the line using the `$P LINE nm` command, then stops current activity with a restart command, `$E LINE nm` .

The commands are defined in the automation policy through the customization dialog panels. The data is stored in the automation control file in the following way:

```
AUTOMATION CONFIGURATION DISPLAY - ENTRY= JES2
TYPE IS $HASP628
CMD          = (PASS1,, 'MVS $P &EHKVAR1')
CMD          = (PASS1,, 'MVS $E &EHKVAR1')
END OF MULTI-LINE MESSAGE GROUP
```

The automation procedure to issue the commands is:

```
/* REXX CLIST to automate $HASP628 */
/* Check whether automation allowed and set TGLOBALs */
'AOCQRY ...'
:
/* Assign EHKVAR1 to parameter 2 (resource name on $HASP628 msg)
then determine whether the first characters are LINE, if not, exit */
ehkvar1 = Msgvar(2)
If Left(ehkvar1,4) <> 'LINE' Then Exit
'GLOBALV PUTT EHKVAR1'
'ACFCMD MSGTYP='Msgid()',SEL=PASS1'
Select
  When rc = 0 Then Nop      /* Command issued OK */
  When rc = 1 Then Do      /* No commands issued; warn if required */
  :
  End
  Otherwise Do             /* Error; perform warning action */
  :
  End
End
Exit
```

Following are the processing steps the automation procedure performs:

1. The `EHKVAR1` variable is assigned the value in the second parameter sent to the automation procedure, which for the `$HASP628` message is the resource type
2. The automation procedure verifies that the resource type is a `LINE`, then sets the variable to a task global variable and calls `ACFCMD`
3. Assuming the second parameter is `LINE21`, two commands are issued from this automation procedure:

```
$P LINE21
$E LINE21.
```

ACFFQRY

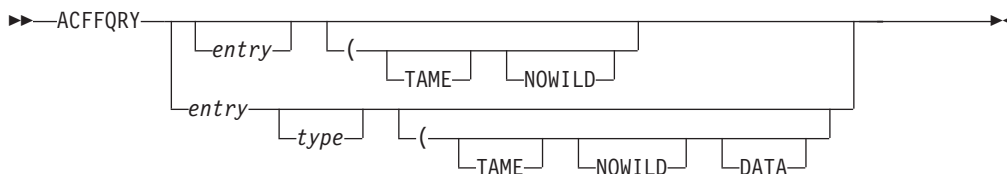
Purpose

The ACFFQRY command provides a fast, pipeable means of accessing the SA z/OS automation control file from your automation procedures.

See also the related command ACF in *IBM Tivoli System Automation for z/OS Operator's Commands*.

Syntax

The following syntax diagram shows how to use the ACFFQRY command to query the automation control file.



Parameters

entry

This is the *entry* value to be used to search the automation control file. The entry value may take the following forms:

- * The entry value is or ends with the wildcard character, unless TAME is specified.

entry

A specific entry value is entered. You must enter a specific entry value if you want to specify a type value.

type

This is the *type* value to be used to search the automation control file. A type value can be specified *only* when a specific entry value is entered. The type value may take the following forms:

- * The type value is or ends with the wildcard character, unless TAME or NOWILD is specified.

type

A specific type value is entered.

TAME

Wildcards in the entry and type name in the automation control file database are to be matched against the entry and type specified on the search. TAME allows for wildcards *in the database* that you are searching. For example, with a constant query string, such as AAA 123 you can match on multiple entries in the automation control file, such as AAA 12*.

This means that if user entries and types have been set up in the automation control file with an asterisk for the last character they are taming candidates. This may be particularly useful for situations where generic rather than specific data is maintained and used in automation procedures.

NOWILD

The asterisk (*) character in the query string is to be treated as a literal.

DATA

The keyword=value data that is related to the entry/type pair is to be returned.

Restrictions and Limitations

A type value can be specified only if a specific entry value is specified.

Usage

It is most efficient if it is called within a PIPE, but may also be called within a TRAP/WAIT/MSGREAD.

Task Global Variables

None.

Messages

Output from ACFFQRY takes the form of a correlated multiline message, with one or two list items and data elements on each line of the message. There are no surrounding message IDs or details.

The first line of the multiline message is always the literal ACFFQRY:, followed by the return code from ACFFQRY. If output is present it begins on line two. This means that output returned in a stem must be processed from element two.

If keyword=data is returned, the entry and type will precede it. Your routines can differentiate entry/type output from data output by the presence of an equals (=) sign. For example:

```
If Pos('=' ,data.n) = 0 then Do
/* data line is an ENTRY TYPE */
End
Else Do
/* data line is an KEYWORD=VALUE */
End
```

- If both entry and type parameters are omitted, a list of all the entries is returned.
- If an entry is specified and the type is omitted, a list of the entry and all the types for that entry is returned.
- If both entry and type are specified, all the data for that entry/type combination is returned.
- If the parameters indicate an area where there is no data, a null list is returned.

Table 3 shows the result for various parameter combinations. An “-” means that an option is irrelevant to the output produced. An asterisk in the DATA column indicates that the keyword=value data is returned.

Table 3. Output from ACFFQRY

Entry	Type	TAME	NOWILD	DATA	Result
		-	-		List of all entries
en* or *		No	No		List of entries starting with “en”.
en* or *		Yes	No		List of all entries starting with en or taming en*.
entry		Yes	-	-	List of entries taming entry
entry		No	-	-	List of types for entry

Table 3. Output from ACFFQRY (continued)

Entry	Type	TAME	NOWILD	DATA	Result
entry	ty*	No	No	*	List of types for entry starting with ty
entry	ty*	Yes	No	*	List of types for entry starting with ty or taming ty*
entry	ty*	No	Yes	*	All data for entry entry and type ty*
entry	ty*	Yes	Yes	*	List of all types for entry taming ty*
entry	type	No	-	-	All data for entry entry and type type
entry	type	Yes	-	*	List of all types for entry taming type

Return Codes

These return codes appear on the first line of the returned data, after the literal ACFFQRY:.

- 0 Data returned.
- 1 There is no data for the specified parameters or SA z/OS is not fully initialized.
- 2 Too many parameters before the opening parentheses. You can specify at most one entry and one type, each of which is a single word.
- 3 Entry/Type combination not allowed. If you have specified an entry including an *, you may not specify a type.
- 5 The SA z/OS global variables containing internal automation control file information have been corrupted.
- 6 You have specified an invalid option.
- 7 You have specified an option more than once.

Examples

Example 1

An ACFFQRY specifying a full ENTRY value only:

```
ACFFQRY SUBSYSTEM
```

This returns all TYPE matches for that ENTRY:

```
ACFFQRY:0
SUBSYSTEM SYSVSSI
SUBSYSTEM SYSVIEW
SUBSYSTEM VLF
SUBSYSTEM LLA
SUBSYSTEM JES
SUBSYSTEM VTAM
SUBSYSTEM TSO
SUBSYSTEM RMF
```

Example 2

An ACFFQRY specifying a full ENTRY value and a full TYPE value:

```
ACFFQRY SUBSYSTEM TSO
```

This returns all keyword=value data that is associated with the ENTRY/TYPE pair:

```
ACFFQRY:0
SUBSYSTEM TSO
JOB=TSO
DESC='Time Sharing Option'
SHUTDLY=00:01:30
```

Example 3

An ACFFQRY specifying a full ENTRY and a wild TYPE:

```
ACFFQRY SUBSYSTEM V*
```

This returns a list of all matching TYPES:

```
ACFFQRY:0
SUBSYSTEM VLF
SUBSYSTEM VTAM
```

Example 4

This example is the same as Example 3, except that the DATA option is specified:

```
ACFFQRY SUBSYSTEM V* (DATA
```

The keyword=value data that is values for all matches are returned:

```
ACFFQRY:0
SUBSYSTEM VLF
DESC='Virt Lib DEF'
SCHEDSUB=MSTR
JOBTYPE=MVS
IPOPTIONS=START
RECYCLEOPT=START
RESTARTOPT=ALWAYS
PARMS=',SUB=MSTR,NN=00'
SHUTDLY=00:03:00
STRTDLY=00:02:00
TERMDLY=00:00:15
JOB=VLF
SUBSYSTEM VTAM
DESC='VTAM V4.1'
PARMS=',,(LIST=FP)'
SHUTDLY=00:01:00
JOB=VTMN24E
```

Example 5

This example shows the use of the TAME option:

```
ACFFQRY CONTROLLER QLN37A07 (TAME
```

All ENTRY/TYPES that include a wildcard that matches the search string are returned:

```
ACFFQRY:0
CONTROLLER QLN*
CONTROLLER QLN37*
CONTROLLER Q*
```

Example 6

This example is the same as example 5 except that the DATA option is specified:

```
ACFFQRY CONTROLLER QLN37A07 (TAME DATA
```

All keyword=value data for the ENTRY/TYPE list is returned:

```
ACFFQRY:0
CONTROLLER QLN*
LOCATION=NEW_YORK
TYPE=LOCAL
OWNER='FRED SMITH'
```

ACFFQRY

```
CONTROLLER QLN37*
LOCATION='Episode 1, Level 3, Oil Refinery'
TYPE=LOCAL
START='MVS VARY 04AE,ONLINE'
OWNER='JIM SMITH'
CONTROLLER Q*
LOCATION=USA
TYPE=GLOBAL
OWNER='BILL SMITH'
```

Example 7

This example shows the result of the following NOWILD option:

```
ACFFQRY CONTROLLER QLN37* (NOWILD
```

The asterisk (*) is treated as a literal in the search pattern:

```
ACFFQRY:0
CONTROLLER QLN37*
LOCATION='Episode 1, Level 3, Oil Refinery'
TYPE=LOCAL
START='MVS VARY 04AE,ONLINE'
OWNER='JIM SMITH'
```

Example 8

The following example shows how to find the job name for a subsystem from a REXX routine, using the NetView PIPE facility:

```
Get_Jobname:
Arg subsystem .
'PIPE NETVIEW ACFFQRY SUBSYSTEM' subsystem '| STEM ALL_DATA.',
'| SEPARATE | LOCATE 1.4 /JOB=/ | TAKE 1 | STEM JOBNAME.'
If all_data.0 < 1 Then
  Say 'PIPE 1 Failed'
If all_data.1 <> 'ACFFQRY:0' Then
  Return
If jobname.0 = 0 Then
  Return subsystem
Parse var jobname.1 'JOB=' jobname .
Return jobname
```

Example 9

This example takes the name of a failing device and finds the appropriate person to notify. It makes use of the TAME option. The data being searched is:

```
DEVFAIL DEV1230,
CONTACT=MIK
DEVFAIL DEV12*,
CONTACT=JB
DEVFAIL DEV34*,
CONTACT=JAQUES
DEVFAIL DEV*,
CONTACT=MIK
CONTACT MIK,
page=00230936473
CONTACT JB,
page=00234628164
CONTACT JAQUES,
page=00237564815
```

The following code fragment takes the number of a failing device and returns the paging number for the person to be notified. Note the use of subroutines that make it easy to write similar queries and could replace the previous example.

```
Get_Page_Num:
Procedure
Arg device_number .
```



```

match = Get_Best_Match('DEVFAIL',device_number)
If match = '' Then
    Return
contact = Get_Key('CONTACT=', 'DEVFAIL',match)
If contact = '' Then
    Return
Return Get_Key('page=', 'CONTACT',contact)

Get_Best_Match:
Procedure
Arg entry ., type .
'PIPE NETVIEW ACFFQRY' entry type '( TAME | STEM DATA.'
If data.0 < 1 Then
    Say 'Get_Best_Match PIPE Failed'
If data.0 <> 'ACFFQRY:0' Then
    Return
match = ''
match_len = 0
Do i = 2 to data.0
    If words(data.i) = 2 Then Do
        data_val = word(data.i,2)
        If Length(data_val) > match_len The Do
            match = data_val
            match_len = Length(match)
        End
    End
End
Return match

Get_Key:
Procedure
Arg key ., entry ., type .
'PIPE NETVIEW ACFFQRY' entry type '(NOWILD | STEM ALL_DATA.',
'| SEPARATE | LOCATE 1.' || length(key) '/' || key || '/',
'| TAKE 1 | STEM DATA.'
If all_data.0 < 1 Then
    Call Terminal_Error 'Get_Key PIPE Failed'
If all_data.1 <> 'ACFFQRY:0' Then
    Return
parse var data.1 .=' data_val
Return data_val

```

ACFREP

Purpose

The ACFREP command allows an automation procedure to issue replies defined in the automation policy. It searches the automation control file for the specified entries, performs variable substitution for predefined variables, then issues the reply.

ACFREP can also issue replies that are built dynamically by the calling automation procedure and passed to ACFREP through a special task global variable named EHKRPY.

ACFREP issues replies to the resource that is identified by the task global variables SUBSAPPL and SUBSTYPE, which are set by the AOCQRY command.

In general you should consider using ISSUEREP or ISSUEACT from the NetView automation table, rather than calling ACFREP directly. This has the following advantages:

- It checks the automation flags for you, to ensure that automation is allowed.

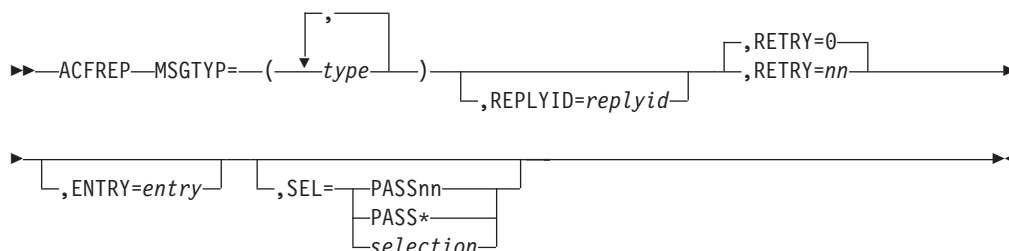
- It checks that the job that issued the message is known to SA z/OS.

If the application responds with a new WTOR after your automation routine issued the ACFREP call, then your automation routine must be invoked again to give SA an opportunity to store new WTOR details. Multiple consecutive ACFREP calls in your automation routine will not find the new outstanding reply and will result in return code 2.

Syntax

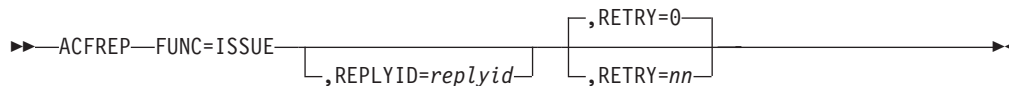
To issue replies directly defined in the automation control file use the following syntax:

1. Syntax for directly defined replies



To issue replies built dynamically by the calling automation procedure use the following syntax:

2. Syntax for dynamically built replies



Parameters

MSGTYP

This value is the message ID in the MESSAGES/USER DATA policy item where the replies to be issued by ACFREP are defined. The value of MSGTYP is typically coded with the message ID or with a generic name such as SPOOLSHORT or SPOOLFULL. The specified type values are searched in the order specified until an entry with the given *entry* and *type* value can be found.

REPLYID

The MVS reply identifier associated with this reply.

This parameter is optional. If it is not specified, the outstanding reply value is retrieved and used, regardless of the specified MSGTYP value.

RETRY

nn specifies the retry count if an outstanding reply is not available. The first time after 1 second and then every two seconds, ACFREP attempts to retrieve an outstanding reply until the retry count is exhausted. When an outstanding reply ID is retrieved, the reply is issued. If no RETRY value is coded, ACFREP defaults to RETRY=0.

ENTRY

This value is the entry in the automation MESSAGES/USER DATA policy item where the replies to be issued are defined. The default is the name of the subsystem that issued the WTOR.

This parameter is mutually exclusive with the FUNC=ISSUE parameter.

SEL

This parameter provides the criteria for the first field in the reply entry. This field gives detailed criteria to select a reply or replies from the automation control file. Based on the MSGTYP, ENTRY and SEL fields, any specific reply can be retrieved from a group of replies associated with a message entry. This parameter is mutually exclusive with the FUNC=ISSUE parameter.

The replies associated with the specified pass selection value defined in the automation policy are issued, along with all replies defined without a selection value. For selection values beginning with PASS, those replies to the pass selection value of PASS* are additionally issued.

If no SEL parameter is coded all replies are selected without respect to any pass selection value in the first field of the reply entry.

PASS nn

PASS nn values can range from 1 through 99 and must be coded without leading zeros, such as PASS1, PASS2, and PASS3.

When SEL=PASS nn is specified, replies associated with the PASS nn selection value defined in the automation policy are issued, along with all replies defined with the selection value of PASS* or with no selection value.

PASS*

When SEL=PASS* is specified, replies associated with the PASS* selection value defined in the automation policy are issued, along with all replies defined without a selection value and all replies defined with a selection value beginning with the prefix PASS.

selection

When SEL=*selection* is specified, the replies associated with the specific selection value defined in the automation policy are issued, along with all replies defined without a selection value.

FUNC=ISSUE

The reply to be issued in response to the incoming WTOR is taken from the task global variable EHKRPY.

This parameter is mutually exclusive with the ENTRY and SEL parameters.

Restrictions and Limitations

The ACFREP command should be called only by an automation procedure or by a command processor. The AOCQRY command must be invoked first to set the SUBSAPPL and SUBSTYPE task global variables.

For serialization reasons, ACFREP must run on:

- The work operator of the subsystem that issued the WTOR if no REPLYID was specified
- The SYSOPER task, if the WTOR was not issued by a subsystem known to SA z/OS

Return Codes

- 0 A reply was found and issued.
- 1 No reply meeting the selection criteria was found.
- 2 No outstanding reply ID was found.
- 3 ACFREP successfully responded to only part of the defined replies.
- 4 Incorrect parameters were used in the call.
- 5 Timeout or other error occurred.
- 6 SA z/OS initialization incomplete, unable to process command request.

Usage

- Consider using ISSUEACT or ISSUEREP from the NetView automation table rather than using ACFREP directly.
- Multiple replies may exist for a given ENTRY, MSGTYP, or SEL field. These replies are processed in the same sequence as they are defined. For the second and subsequent replies, ACFREP always retrieves the outstanding reply number of a subsystem before issuing the reply. If an outstanding reply number does not exist when the reply should be issued, ACFREP attempts a retry if so defined. Retries may be defined either through the RETRY keyword of ACFREP or through the retry value specified in the policy entry. The retry value that you specified in the RETRY keyword takes precedence over that in your policy if both are specified. There is a 2-second delay between retry attempts.
- SA z/OS variable EHKRPYHI is returned to the calling automation procedure as a task global variable value. ACFREP retrieves all reply entries for a given ENTRY or MSGTYP value, searches for the highest PASS nn number, and returns it in the variable EHKRPYHI. You can use this number to determine whether all available commands are issued and an appropriate error message is issued to the operator. If PASS nn is not coded, EHKRPYHI is zero.
- Variables are available to change the reply entered in the automation control file. Variables EHKVAR0 through EHKVAR9 and EHKVART must be defined as task global variables in the calling automation procedure and must be initialized with the data to change the replies. These variables are passed to the ACFREP command. Whenever ACFREP finds a detail reply entry in the automation control file, it scans the reply entry looking for &EHKVAR n . If an EHKVAR n variable is found, the value stored in the variable replaces the &EHKVAR n in the reply entry. You can code multiple &EHKVAR n variables in a single reply entry. Delimiters are unnecessary, and you can code the variables between any other text.
- If your automation procedure issues a TRAP command, you must save the message variables upon entry, because this information is lost whenever message processing is started.

Task Global Variables

The following run time variables are used by ACFREP for different purposes:

EHKRPY

When ACFREP is called with FUNC=ISSUE, this variable must provide the reply that is to be issued.

EHKRPYHI

This variable contains the number of the highest PASS nn selection for defined replies to the specified entry and type in the automation policy.

EHKRPYTEXT

The text for the AOF570I confirmation message, after having issued the reply.

EHKVAR0 through EHKVAR9 and EHKVART

These variables can be used to alter reply definitions in the automation policy. If included in the defined replies with a leading &, the variables are substituted by their values before the replies are issued. The values of these variables must be provided by the calling automation routine.

&APPLPARMS

This variable provides the value that is entered in the APPLPARMS parameter of the INGREQ command when starting or stopping the resource related to the specified entry. The value of this variable is only available during startup or shutdown processing of a resource under the control of SA z/OS and can be used to alter replies entered in the automation policy.

If the AOCQRY command has been invoked in the calling automation routine, it sets the following task global variables if the appropriate information is available (where *x* can be either S for subsystem, or P for parent subsystem):

- SUB x APPL
- SUB x ASID
- SUB x CATEGORY
- SUB x CMDPFX
- SUB x DESC
- SUB x FILE
- SUB x FILTER
- SUB x IPSTACK
- SUB x JOB
- SUB x PATH
- SUB x PID
- SUB x PLEX
- SUB x PORT
- SUB x PROC
- SUB x SCHEDSS
- SUB x SHUTDLY
- SUB x SPARM
- SUB x SUBCAT
- SUB x SUBID
- SUB x SUBTYPE
- SUB x SYMBOL n ($n=1-9$)
- SUB x USER
- SUB x USSJOB

These task global variables are substituted when they are specified with a leading & in replies that have been defined in the automation policy or provided in the task global variable EHKRPY. Refer to the tables in “Task Global Variables” on page 44 for AOCQRY.

Valid symbols for provider run time variables are as follows (where *x* can be either S for subsystem, or P for parent subsystem):

- SU2 x APPL
- SU2 x ASID
- SU2 x CATEGORY
- SU2 x CMDPFX
- SU2 x DESC
- SU2 x FILE
- SU2 x FILTER
- SU2 x IPSTACK
- SU2 x JOB
- SU2 x PATH

ACFREP

- SU2xPID
- SU2xPORT
- SU2xPROC
- SU2xSCHEDSS
- SU2xSHUTDLY
- SU2xSUBCAT
- SU2xSUBID
- SU2xSUBTYPE
- SU2xSYMBOL n ($n=1-9$)
- SU2xUSER
- SU2xUSSJOB

Common Global Variables

Common global variable AOFJESPREFIX is substituted in the reply to be issued when found.

Examples

Example 1

This example shows the relationship between ACFREP and automation policy. The message to automate, \$HASP426, is produced by the JES2 subsystem, requesting the JES2 startup specifications. The automation procedure responds to this by calling ACFREP to issue a reply of WARM,NOREQ from the automation control file.

The data is stored in the automation control file in the following way:

```
AOFK3D0X          SA z/OS - Command Response          Line 1    of 4
Domain ID   = IPSNO   ----- DISPACF -----      Date = 06/06/00
Operator ID = NETOP1                                     Time = 13:30:53

Command = ACF ENTRY=JES2,TYPE=$HASP426,REQ=DISP
SYSTEM = KEY3      AUTOMATION CONFIGURATION DISPLAY - ENTRY= JES2
-----
AUTOMATION CONFIGURATION DISPLAY - ENTRY= JES2
TYPE IS $HASP426
REPLY      = (.,'WARM,NOREQ')
END OF MULTI-LINE MESSAGE GROUP
```

The automation procedure to issue this reply is:

```
/* REXX CLIST to automate the reply to $HASP426          */
/* Check whether automation allowed and set TGLOBALs     */
'AOCQRY ...'
:
:
'ACFREP MSGTYP=$HASP426,REPLYID='Replyid()',ENTRY=JES2'
Select
  When rc = 0 Then Nop          /* Reply issued OK          */
  When rc = 1 Then Do          /* No reply issued; warn if required */
  :
  End
  Otherwise Do                  /* Error; perform warning action */
  :
  End
End
Exit
```

ACFREP uses the parameters that are passed to the routine to find corresponding entries in the automation control file. Because no SEL parameter is coded, no selection restriction is made concerning the first field of the command entry.

Note that the function Replyid() is used for the REPLYID parameter. This function is a standard NetView REXX function that will only return a value to an automation procedure called from the NetView automation table, and only if a reply is required. You can use this value when calling ACFREP.

Upon return to the automation procedure, the rc special variable is checked to ensure that a reply was found in the automation control file. The automation procedure takes appropriate action if a reply is not found or a processing error occurs in ACFREP.

Note: Assuming that AOCQRY was checking the Start automation flag, this example routine could be replaced by coding:

```
ISSUEREP AUTOTYP=START
```

Example 2

This example uses the same scenario as Example 1, but shows how you can use the defaults to minimize coding. The message to automate, \$HASP426, is produced by the JES2 subsystem and requests the JES2 startup specifications. The automation procedure responds to this by calling ACFREP to issue a reply of WARM,NOREQ from the automation control file.

The reply is defined in the automation policy in the same way as Example 1.

The automation procedure to issue the reply is:

```
/* REXX CLIST to automate the reply to $HASP426          */
/* Check whether automation allowed and set TGLOBALs    */
'AOCQRY ...'
:
'ACFREP MSGTYP='Msgid()',REPLYID='Replyid()'
Select
  When rc = 0 Then Nop      /* Reply issued OK          */
  When rc = 1 Then Do      /* No reply issued; warn if required */
:
  End
  Otherwise Do             /* Error; perform warning action */
:
  End
End
Exit
```

This example differs from Example 1 in the following ways:

- ACFREP uses a NetView REXX function for the MSGTYP field and assumes the defaults for the ENTRY and SEL fields.
The ENTRY field defaults to the value of SUBSAPPL. AOCQRY will set this value to the name of the application that AOCQRY was invoked with. In this case the value is JES2.
- The MSGTYP field uses the NetView REXX function Msgid(), which contains the message identifier for the message that called the automation procedure. This message identifier is supplied only to an automation procedure called from the NetView automation table. Use this value when calling ACFREP. Note that calling WAIT will replace the value of Msgid().

Note: Assuming AOCQRY was checking the Start flag, this example could be replaced with:

```
ISSUEREP AUTOTYP=START
```

Example 3

This example shows the use of *PASS_n* logic in an automation procedure. The message to automate, \$HASP098, is produced by the JES2 subsystem and requests the JES2 shutdown options. The automation procedure responds to this, the first Reply time, by calling ACFREP to issue a REPLY of DUMP from the automation control file, and the second time by issuing a reply of PURG.

Reply information is defined in the automation policy through the customization dialogs. The data is stored in the automation control file in the following way:

```

AOFK3D0X          SA z/OS - Command Response      Line 1    of 5
Domain ID   = IPSNO   ----- DISPACF -----   Date = 06/06/00
Operator ID = AFRANCK                               Time = 13:36:31

Command = ACF ENTRY=JES2,TYPE=$HASP098,REQ=DISP
SYSTEM = KEY3      AUTOMATION CONFIGURATION DISPLAY - ENTRY= JES2
-----
AUTOMATION CONFIGURATION DISPLAY - ENTRY= JES2
TYPE IS $HASP098
REPLY          = (PASS1,, 'DUMP')
REPLY          = (PASS2,, 'PURG')
END OF MULTI-LINE MESSAGE GROUP

```

The automation procedure to issue these replies is:

```

/* REXX CLIST to automate $HASP098                               */
/* Check whether automation allowed and set TGLOBALs            */
'AOCQRY ...'                                                    */
:
/* Increase the counter unique to this automation procedure     */
'GLOBALV GETC HASP098_CNT'                                       */
If hasp098_cnt = " Then hasp098_cnt = 1
Else hasp098_cnt = hasp098_cnt + 1
'GLOBALV PUTC HASP098_CNT'
/* Issue the ACF reply for the pass number as determined       */
'ACFREP MSGTYP='Msgid()',REPLYID='Replyid()',SEL=PASS'hasp098_cnt
Select
  When rc = 0 Then Nop      /* Reply issued OK                    */
  When rc = 1 Then Do      /* No reply issued; warn if required */
:
  End
  Otherwise Do             /* Error; perform warning action    */
:
  End
End
Exit

```

This example differs from the previous examples in the following ways:

- The automation procedure uses a unique common global variable, in this case HASP098_CNT, to maintain a PASS counter. The automation procedure adds 1 to this counter each time it is processed, then appends the counter to the SEL=PASS field. During processing, the counter is translated, and PASS1 or PASS2 is run. Note that a null test is required to set the counter to 1 if it has not been set before. If the counter exceeds 2 then the ACFREP will set a return code of 1 since there is no matching entry in the automation control file.
- Another automation procedure that resets the counter is necessary to complete the logic flow. In this example, this automation procedure is processed when the final JES2 message or a startup message is received.

The automation procedure to reset the counter is:


```

/* REXX CLIST to reset the counter                               */
hasp098_cnt = ''
'GLOBALV PUTC HASP098_CNT'
Exit

```

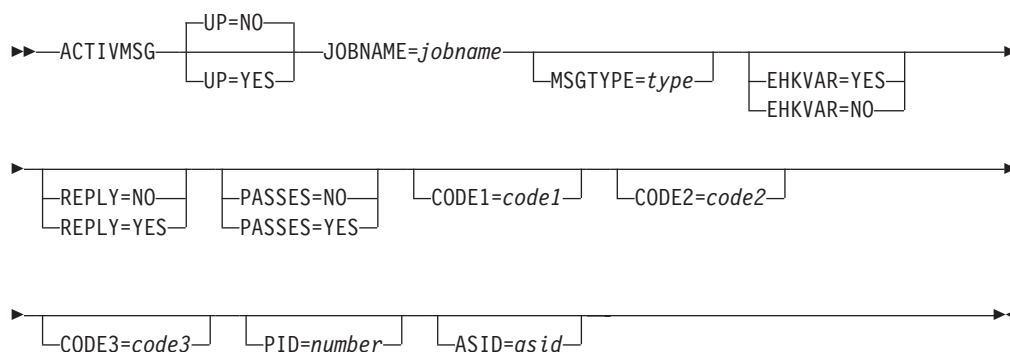
Note: To ensure serialization of access to the NetView global dictionary and the correct ordering of the replies issued, the NetView automation table entry should route the command to a specific operator if the message may occur more than once in quick succession.

ACTIVMSG

Purpose

You can use the ACTIVMSG command to respond to ACTIVE and UP messages from an application by changing the SA z/OS status of the application. ACTIVMSG calls the ISSUEACT command to also issue commands and replies that are defined in the automation policy for both the ID of the ACTIVE or UP message and for the new status of the application. Typically, ACTIVMSG is called from the NetView automation table.

Syntax



Parameters

UP This parameter is used to distinguish between ACTIVE messages and UP messages. ACTIVE messages indicate that the job associated with an application is working but is not yet available for use. UP messages indicate that the job associated with an application is available for use.

NO NO should be used if you are responding to an application ACTIVE message. The application is placed in ACTIVE status if it is not there already. UP=NO is the default.

YES

YES should be used if you are responding to an application UP message. The application is placed in UP status if it is not there already. If the application is a transient job then it is placed in RUNNING status.

JOBNAME

The name of the job that the message is for. If not specified, the job name is taken from the message's job name field. You must supply a value for the job name if you are calling ACTIVMSG from a CLIST.

MSGTYPE

This parameter is used to search for command and reply entries to *subsystem/msgtype*-pairs in the automation control file, where *subsystem* is the subsystem name derived from the job name.

When a match occurs, the commands that are associated with the entries are issued. This is in addition to the entries that are associated with the ENTRY-TYPE pair *subsystem/ACTIVE* if UP=NO and *subsystem/UP* if UP=YES.

If parameter MSGTYPE is not specified, the message identifier of the message that ACTIVMSG is called for is taken as the default.

EHKVAR

This parameter determines whether the tokens of the parsed message text are to be stored in task global variables EHKVAR0 through EHKVAR9 and EHKVART.

YES

The tokens of the triggering message are to be assigned to the task global variables EHKVAR*n*.

NO No values are to be assigned to the task global variables EHKVAR*n*.

REPLY

This parameter determines whether a defined reply is issued for a message that ACTIVMSG has been called for.

YES

A defined reply in the automation policy for the message that is being handled by ACTIVMSG is issued. REPLY=YES is assumed as the default if the message is a WTOR, otherwise the default is REPLY=NO.

NO A defined reply for a WTOR that is being handled by ACTIVMSG is not issued.

PASSES

Specifies whether passes are used to issue commands or replies (or both) that have been defined in the automation policy.

YES

PASSES=YES is passed to the ISSUEACT command.

NO PASSES=NO is passed to the ISSUEACT command.

CODE1=code1 CODE2=code2 CODE3=code3

These parameters are passed to the ISSUEACT command, where they are used to select defined commands and replies via code entries.

PID

The process ID of the resource. Together with the ASID, it uniquely identifies the resource.

ASID

The ASID that is associated with the resource. Together with the PID, it uniquely identifies the resource.

Restrictions and Limitations

- If ACTIVMSG is driven by a delete operator message, no action is taken in response to this message.
- Defined commands and replies are only issued in response to a message or a status change if the start flag of the related minor resources of the application allows automation.

Usage

You should typically call the ACTIVMSG command from the NetView automation table.

It is recommended that you use ACTIVMSG for all IEF403I (job started) messages.

If ACTIVMSG is called for a WTOR and it is not replied to, OUTREP is called to track the WTOR.

If you are invoking ACTIVMSG for a generic message you should use the ING\$QRY NetView automation table function to screen the message before invoking ACTIVMSG. See Chapter 4, "ING\$QRY NetView Automation Table Function," on page 187 for more information.

ACTIVMSG should run on the working operator of the subsystem that issued the message. Otherwise, the ACTIVMSG command will run asynchronously to the calling procedure. This means that when the calling procedure regains control, the status of the affected subsystem may not yet have changed.

All commands and replies that are triggered through ACTIVMSG have access to the SAFE, called AOFMSAFE, that stores the message that caused the ACTIVMSG call.

Task Global Variables

EHKVAR0 through EHKVAR9 and EHKVART

When defining the commands in the automation control file to be issued by command ACTIVMSG, the variables &EHKVAR0 through &EHKVAR9 and &EHKVART can be used to be substituted by the tokens of the parsed message that has driven ACTIVMSG. &EHKVAR0 will be substituted by the message ID, &EHKVAR1 by the first token of the message text after the message ID, &EHKVAR2 with the second token and so forth. &EHKVART will be substituted by the trailing message text after the 9th token.

Examples

The following example shows how ACTIVMSG is called from the NetView automation table:

```
IF MSGID = 'IEF403I' & TOKEN(2) = SVJOB & DOMAINID = %AOFDOM%
  & ATF('ING$QRY APPL,,JOB='VALUE(SVJOB)) ^= ''
THEN
EXEC(CMD('ACTIVMSG JOBNAME=' SVJOB)
ROUTE(ONE %AOFOPGSSOPER%));
```

AOCFILT

Purpose

The AOCFILT command is used to screen messages that invoke other commands. Although it adds to the overhead of a useful invocation of a command, it greatly reduces CPU used to detect an unnecessary invocation.

Note: For performance reasons consider using ING\$QRY instead, see Chapter 4, "ING\$QRY NetView Automation Table Function," on page 187.

Syntax

```

  >> AOCFILT jobname command >>
        |_____|
        *
  
```

Parameters

jobname

This is the name of the job that the message refers to. If an * is specified the default job name for the message, retrieved with the NetView `jobname()` function, is checked.

command

This command is issued (in a PIPE) if the *jobname* parameter is the name of a job known to SA z/OS. If the job name is not the name of a job of a SA z/OS-controlled application, the command is not issued.

Restrictions and Limitations

- The command should be invoked only when there is a message in the default safe. Normally this will be from the NetView automation table (AT).
- You must obtain the job name before you invoke AOCFILT.

Return Codes

AOCFILT produces a return code of 0.

Usage

You should code the AOCFILT command in the NetView automation table (AT) where you are using a generic message (such as IEF403I) to invoke one of the SA z/OS commands (such as ACTIVMSG).

AOCFILT routes the command that is passed to it to the auto operator that is responsible for that particular subsystem.

AOCFILT is not as efficient as explicitly screening for the message in the AT, but may be more efficient than negative screening. AOCFILT also makes the automation statement more portable, in that you do not have to update it if you define a new application to SA z/OS.

Examples

In the following example, the NetView automation table is used to block out all IEF403I messages concerning jobs starting with the letters BAT, and AOCFILT is used to screen the other IEF403I messages:

```

IF MSGID = 'IEF' . & DOMAINID = %AOFDOM% THEN BEGIN;
...
  IF MSGID = 'IEF403I' THEN BEGIN;

    IF TOKEN(2) = 'BAT' . THEN DISPLAY(N) NETLOG(Y);

    IF TOKEN(2) = SVJOB THEN
      EXEC(CMD('AOCFILT ' SVJOB ' ACTIVMSG JOBNAME=' SVJOB)
        ROUTE(ONE %AOFOPGSSOPER%));
    END;
  ...
  ALWAYS;
END;
  
```

Related Commands

- “ACTIVMSG” on page 29
- “HALTMSG” on page 74
- “ISSUEACT (ISSUECMD, ISSUEREP)” on page 161
- “TERMMSG” on page 171

AOCGETCN

Purpose

The AOCGETCN command obtains an extended MCS console with a unique name for an operator or autotask issuing the command. If an MVS console is already associated with that task, it is released.

The default console name is the character A, followed by the last 5 characters of the task name concatenated with the last two characters of the system name.

Syntax

►►—AOCGETCN—*parameters*—►►

Parameters

Optionally, you may supply one or more parameters that are valid for NetView's GETCONID, for example, ALERTPCT, MIGRATE, QLIMIT, QRESUME, or STORAGE.

If you specify more than one parameter, you can either separate them by blank or by comma, for example:

```
AOCGETCN MIGRATE=YES,STORAGE=1000
```

For further information and a list of valid GETCONID parameters and their descriptions, refer to the NetView documentation.

Restrictions and Limitations

The previous console will be released even if AOCGETCN fails to obtain the new console.

The GETCONID parameters `CONSOLE=xxxxxxx` and `AUTH=yyyyyy` are not supported. If you enter them, they will be ignored.

Usage

Console names within a sysplex must be unique. The task name is used if the console name is not specified. To avoid possible naming conflicts due to common task names AOCGETCN should be used to obtain a console with a unique name. The characters that are used in determining the unique console name can be tailored by updating the common global variable AOFNPMASK. Refer to *IBM Tivoli System Automation for z/OS Customizing and Programming* for further information.

AOCGETCN

Example

Issue AOCGETCN, for example, in the initial clist of an operator or autotask. As soon as an MVS command is issued by the task, a console is allocated with the console name that has been set by AOCGETCN.

The default name for the console is the character A, followed by the last 5 characters of the task name concatenated with the last two characters of the system name.

Thus, task OPER1 on system FOC1 obtains the default extended console name OPER1. When the AOCGETCN command is issued, the console AOPER1C1 is now associated with OPER1.

AOCMSG

Purpose

The AOCMSG command displays and logs messages. AOCMSG merges variable data specified as parameter values in the AOCMSG call with fixed message text to produce an SA z/OS message. You can display the resulting message on a NetView console and log it in the NetView log.

The message format depends on the message ID and variable data placed in the message.

If you specify one or more message classes in the message, AOCMSG also performs message class matching and sends the message as a notification message to one or more notification operators defined to receive those classes of notification messages.

AOCMSG uses the NetView message handling facilities, specifically NetView macros DSIMDS and DSIMBS. When you want to define user messages you must code a message definition module named AOFM aaa where aaa is the message prefix. Refer to *NetView Customization: Using Assembler* for the coding. Examples 1 and 2 in this section require a message definition module of AOFMABC.

The parsing within AOFMSG has been rewritten with an SA z/OS parsing routine used instead of DSIPRS. This allows SA z/OS to be more flexible in the handling of parameters. The parsing rules are:

- The only delimiter recognized in parsing the command is the comma.
- Tokens surrounded by single quotation marks will be stored without the quotation marks.
- A token containing two consecutive single quotation marks will be stored with only one of the quotation marks.
- Leading and trailing spaces are removed except that spaces inside quotation marks are not removed.
- Instead of rejecting a command with mismatched quotation marks an attempt is made to break the command into tokens.

The rules are illustrated by the following examples:

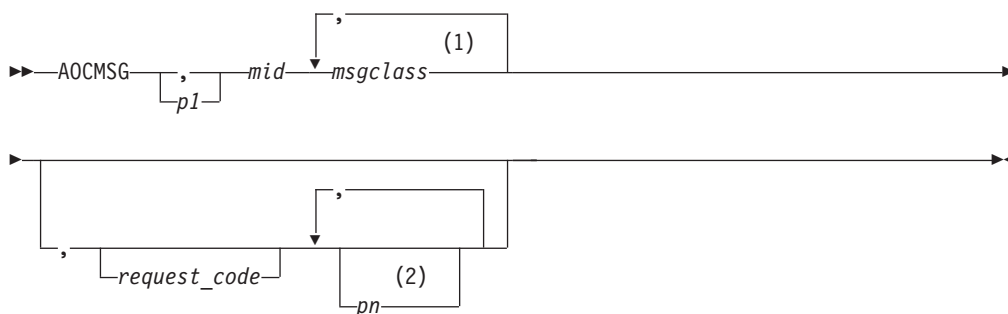
COMMAND	TOKENS
A,BCDEF, G	(A) (BCDEF) (G)
'A B C' , ' EF GH'	(A B C) (EF GH)
ABC,,DEF	(ABC) () (DEF)
'ABC,DEF,GHI	(ABC,DEF,GHI)

'ABC' 'DEF'	(ABC'DEF)
'ABC,DEF	('ABC) (DEF)
ABC'DEF	(ABC'DEF)
ABC'DEF'	(ABC'DEF')
'ABC'DEF	(ABC) (DEF)
'ABC'DEF'	(ABC) (DEF')
'ABC' 'DEF	(ABC) ('DEF)
ABC''	(ABC')

Note: AOCMSG has the facility to use MVS descriptor codes to control the message flow at the master console. Refer to *IBM Tivoli System Automation for z/OS Messages and Codes* for a table of message types and descriptor codes used by AOCMSG.

Syntax

Parameters are positional.



Notes:

- 1 Up to 10 optional message classes can be specified with the *mid* parameter. If used, message classes should be separated from the *mid* value and each other by at least **one** blank.
- 2 Parameters 2–9 may be specified here. Parameters are positional, so non-specified parameters must be represented by a comma.

Parameters

p1–p9

These are parameter values that are substituted into the message text (located in a NetView DSIMSG member) in place of NetView message variables &1 through &9, respectively. These parameter values are all optional. However, because parameters are positional, if you do not specify *p1*, you must code a comma for that parameter position, for example:

```
aocmsg ,abc123,,date(),time()
```

mid

The message ID to be issued. This parameter is required. The message ID must be a valid message installed in the NetView message library, that is, in data set members identified in DSIMSG. The message ID can be specified in the following ways:

- A 3-digit number, which is assumed to have a prefix of AOF.

This message ID value relates to a message in DSIMSG member DSIAOF nn . For example, a message ID value of 203 is for SA z/OS message AOF203I, which is in DSIMSG member DSIAOF20.

- A 6-digit ID consisting of a 3-character prefix followed by a 3-digit message ID number. The first character of the prefix must be alphabetic.

This message ID value relates to a message in DSIMSG member DSI $xxxnn$, where xxx is the prefix value and nn is the first two digits of the message ID number. For example, a message ID value of ABC123 is for message ABC123I, which is in DSIMSG member DSIABC12.

- A 7-character ID consisting of a 4-character prefix followed by a 3-digit message ID number. The first character of the prefix must be alphabetic.

The primary use for this type of message ID format is when coding a message ID for a message that has a 4-digit prefix.

When this type of message ID value is specified, AOCMSG drops the third of the four prefix characters to create the string used for searching DSIMSG members and retrieving the desired message. The actual message issued uses all four prefix characters.

For example, a message ID value of ABCD123 is used to retrieve message ABCD123I, which is in DSIMSG member DSIABD12 (note that the C is dropped in the DSIABD12 member name).

The message ID value can be up to 7 characters long.

Up to 10 optional dynamic message classes can be specified through the mid field. If specified, optional message classes will be merged with the message classes defined in the message member (if there are any) up to a maximum of 10 message classes. If the total number of message classes exceeds 10, those specified on the AOCMSG call will take precedence over those specified in the message member.

The rules for dynamic message classes are the same as for those defined in the message member.

Message classes specified on the AOCMSG call are taken into consideration for the following *request_codes*:

blank
LOG
MIM

When NOMID is used, any message classes specified on the AOCMSG call will be ignored. However, any message classes defined in the message member will continue to appear in the resulting message text.

request_code

This parameter specifies the type of message processing request the AOCMSG command performs. The value for this parameter can be one of the following:

blank

If you leave this parameter position blank, or enter any text other than the values listed below, AOCMSG will generate the message for display. This is the default.

LOG

AOCMSG generates the message and logs the message in the NetView log instead of displaying it on the issuer's console.

MIM (Message in Message)

AOCMSG generates the message and strips the message ID value (*mid*) from the generated message, leaving only the message text. The first word

in the message is treated as a valid message ID value (*mid*), and processing continues as if that word were the original *mid*. That is, AOCMSG performs message class matching and notification. See the AOCMSG examples for an example of how this parameter value affects the issued message.

NOMID (No Message ID)

AOCMSG generates the message and strips the message ID value (*mid*) from the generated message, leaving only the message text. AOCMSG does not perform message class matching and notification. See "Example 2" on page 38 for an example of how this parameter value affects the issued message.

Note: With the exception of the *NOMID request_code* value, forwarding of notification messages to notification operators occurs regardless of the value specified for this parameter.

Restrictions and Limitations

Each variable parameter value besides the message ID value (*mid*) can be up to 80 characters long, but the total maximum message length is 470 characters.

An operator can call the AOCMSG command from an automation procedure or command processor, or issue it directly from a display station.

Return Codes

0 AOCMSG processed normally.

>0 and <60

An error occurred while processing the NetView DSIPSS macro. The return code is actually from DSIPSS.

60 An error occurred while processing the NetView DSIGET macro to request storage. No storage space is available.

>60

An error occurred while processing the NetView DSIPRS macro.

Note: If you receive return codes other than 0 and 60, refer to *NetView Customization: Using Assembler* for information on resolving the NetView macro problems.

Error messages returned by AOCMSG are:

AOF262E MESSAGE ID *mid* INVALID, MUST BE "NNN", "ABCNNN", OR "ABCDNNN".

AOF263I MESSAGE ID NUMERIC "*nnn*" IS NOT NUMERIC.

AOF264I TOO FEW PARAMETERS ON AOCMSG COMMAND, 2 IS MINIMUM.

abc000I USER MESSAGE *mid* ISSUED BUT DOES NOT EXIST IN MESSAGE TABLE
DSIabcnn - CALL IGNORED.

Note: In message *abc000I*, the variable *abc* represents the product identifier portion of the message ID.

Usage

- Parameter values passed to AOCMSG depend on the format of the message entry as coded in the DSIMSG member *DSIxxxxnn*.

- AOCMSG uses NetView message handling facilities, DSIMDS and DSIMBS in particular. Refer to *NetView Customization* for details about using DSIMDS to create your own messages.
- AOCMSG implements the SA z/OS notification message function to allow you to forward messages to notification operators. This aspect of AOCMSG processing can be useful if you develop new messages and want notification operators to receive them.

The notification message function is implemented by assigning message classes to your messages. Message classes are assigned within the text of the messages in the DSIMSG member (DSI:xxxm). In the text for the message, specify the class or classes (up to five) after the message ID number and before the message text. For example, the following entry for a message assigns message classes 10 and 40 to the message. The message will be issued as a notification message to any notification operators defined to receive class 10 or 40 messages.

```
123I 10 40 THE EAGLE HAS &1
```

Examples

Example 1

Entries for messages in DSIMSG member DSIABC12 are as follows:

```
*****
120I ...
121I ...
122I &1 &2 ON THE &3
123I 10 40 THE EAGLE HAS &1
124I ...
*****
```

An automation procedure contains the following AOCMSG calls referencing messages ABC122 and ABC123.

```
<other automation procedure code>
:
AOCMSG HELP,ABC122,,IS,WAY
AOCMSG LANDED,ABC123
:
<other automation procedure code>
```

When AOCMSG is called as specified in the automation procedure, DSIMSG member DSIABC12 is searched for messages ABC122I and ABC123I. Variable substitution for the variables in the message entries occurs, resulting in the following messages being generated:

```
ABC122I HELP IS ON THE WAY
ABC123I THE EAGLE HAS LANDED
```

Note: Because the DSIMSG member entry for ABC122I does not specify message class information, only the issuer of the automation procedure receives the message, not any notification operators. Because the DSIMSG member entry for message ABC123I specifies message classes 10 and 40, notification operators defined to receive message classes 10 and 40 also receive message ABC123I.

Example 2

Use of the AOCMSG *request_code* parameter value NOMID has the following effect on the messages generated.

The same entries in DSIMSG member DSIABC12 are used.

The AOCMSG calls using the NOMID *request_code* parameter value are as follows:

```
<other automation procedure code>
:
AOCMSG HELP,ABC122,NOMID,IS,WAY
AOCMSG LANDED,ABC123,NOMID
:
<other automation procedure code>
```

These calls and the DSIABC12 entries result in the following messages:

```
HELP IS ON THE WAY
10 40 THE EAGLE HAS LANDED
```

Note: In message ABC123I, the message classes 10 and 40 have not been processed as message classes and appear in the message text. No notification operators receive either message. This is an error for message ABC123. The message is not implemented to use the NOMID parameter value effectively.

Use of the AOCMSG *request_code* parameter value MIM has the following effect on the messages generated.

The same entries in DSIMSG member DSIABC12 are used.

The AOCMSG calls using the MIM *request_code* are as follows:

```
<other automation procedure code>
:
AOCMSG HELP,ABC122,MIM,IS,WAY
AOCMSG 'HELP 40',ABC122,MIM,IS,WAY
AOCMSG LANDED,ABC123,MIM
:
<other automation procedure code>
```

These calls and the DSIABC12 entries result in the following three messages:

HELP IS ON THE WAY

The text HELP is considered to be the new message ID. Because no message classes are in the AOCMSG call, no notification operators receive the message.

HELP IS ON THE WAY

In this case, the value 40 is processed as a message class. This processing causes notification operators defined to receive class 40 messages to also receive this message.

10 THE EAGLE HAS LANDED

The value 40 is processed as a message class, as in previous AOCMSG examples. In contrast, the value 10 is processed as the message ID, not a message class. Message ABC123 is not implemented to effectively use the MIM parameter value.

AOCQRES

Purpose

The AOCQRES command examines and returns information about where a resource resides in a sysplex. Optionally, AOCQRES also tries to obtain status information on resources.

Syntax

```

  >> AOCQRES subsystem_name [(-STATUS)]
  
```

Parameters

subsystem_name

Specifies the name of the subsystem.

- * This causes the command to return information about all subsystems within the sysplex.

STATUS

If you specify STATUS, another column will be added to the output. This column contains the current automation status of each subsystem.

Return Codes

- 0 The AOCQRES command completed successfully.
- 1 An error occurred while processing the AOCQRES command. See the accompanying message for the cause of the error.
- 2 The specified subsystem is either unknown or currently not registered.

Usage

The command is to be used within a NetView PIPE statement.

Examples

When you issue the command:

```
PIPE NETV AOCQRES TSO (STATUS | STEM ABC.
```

within a REXX procedure and the system where this command is issued is in a sysplex with four systems, the stem variable ABC. will be assigned something similar to the following:

```

ABC.0 = 4
ABC.1 = TSO      TSO      KEY3      UP
ABC.2 = TSO      TSO      KEY4      AUTODOWN
ABC.3 = TSO      TSO      KEY5      STARTED
ABC.4 = TSO      TSO      KEY6      RESTART
  
```

The first token of the data is the subsystem name, the second token is the subsystem's job name, the third token is the system name and the last token is the subsystem's status.

AOCQRY

Purpose

The AOCQRY command verifies that automation is allowed for a specific resource. AOCQRY does the following:

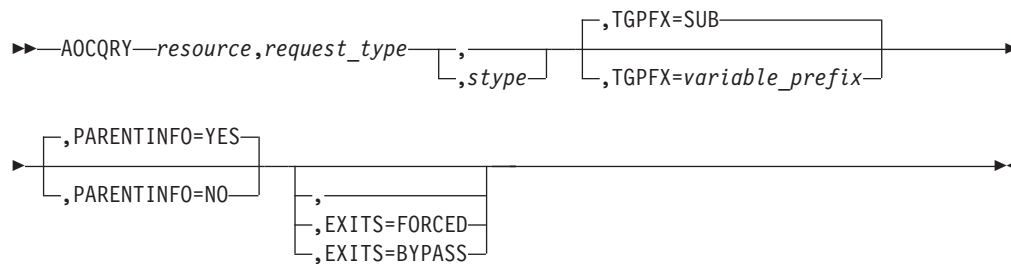
- Searches the automation policy to verify that the resource is defined to SA z/OS
- Checks that the automation flags for that resource allow automation
- Initializes certain control variables for use by the calling automation procedure
- Drives automation flag exits

- Initializes AOCQRY task global variables with application information

A call to AOCQRY is intended to be a standard component of most automation procedures. AOCQRY should be called whenever resource automation is required to verify whether automation should continue.

AOCQRY only works for applications that have been defined to automation using the application policy object of the customization dialogs.

Syntax



Parameters

resource

The resource name that automation should be checked for. This value can be a:

- Job name
- Subsystem name
- Subsystem minor resource, such as CICS®.TRAN.APPL1, where the first qualifier can be the job name of the subsystem
- MVS component minor resource, such as MVSESA.SMF
- SUBSYSTEM
- DEFAULTS

If *stype* is coded, the resource is assumed to be a minor resource and, if the specified resource name does not already begin with the value of *stype*, it is prefixed with the value of *stype*, separated by a period.

If *stype* does not specify the name of a subsystem or MVSESA (the value of the AOFSYSTEM common global variable), the resource name is prefixed with MVSESA. When, for example, calling AOCQRY SMF RECOVERY MVSESA or AOCQRY MVSESA.SMF RECOVERY MVSESA, MVSESA.SMF is assumed as the resource name in both cases.

This parameter is required.

request_type

The type of automation checks and information retrieval functions AOCQRY performs. *request_type* is required and must be one of the following:

AUTOMATION

Only the Automation flag is checked to determine whether automation is allowed. Data retrieval from the automation policy occurs as described for the CFGINFO option. If the *stype* parameter is coded data retrieval does not occur.

INITSTART

The Automation flag and the Initstart flag are checked in determining

whether automation is allowed. Data retrieval from the automation policy occurs as described for the CFGINFO option. If the *stype* parameter is coded data retrieval does not occur.

START

The Automation flag and the Start flag are checked in determining whether automation is allowed. Data retrieval from the automation policy occurs as described for the CFGINFO option. If the *stype* parameter is coded data retrieval does not occur.

RECOVERY

The Automation flag and the Recovery flag are checked in determining whether automation is allowed. Data retrieval from the automation policy occurs as described for the CFGINFO option. If the *stype* parameter is coded data retrieval does not occur.

TERMINATE

The Automation flag and Terminate flag are checked in determining whether automation is allowed. Data retrieval from the automation policy occurs as described for the CFGINFO option. If the *stype* parameter is coded data retrieval does not occur.

RESTART

The Automation flag and the Restart flag are checked in determining whether automation is allowed. Data retrieval from the automation policy occurs as described for the CFGINFO option. If the *stype* parameter is coded data retrieval does not occur.

CFGINFO or STATUS

Selected information about the specified resource is retrieved from the automation policy and provided in predefined task global variables. Regardless of whether the resource is a subsystem name or a minor resource to a subsystem, information retrieval is performed for the subsystem and its parent. This parameter is mutually exclusive with the *stype* parameter and is not supported for the specified resources DEFAULTS, SUBSYSTEM or MVSESA (the value of the AOFSYSTEM common global variable) and its minor resources. Supported task global variables are described in "Task Global Variables" on page 44.

CFGONLY or CFG-ONLY or STATUS-ONLY

Selected information is provided as described for the CFGINFO option, but without information to the subsystem parent.

stype

If *stype* is coded, the specified resource parameter is assumed to be a minor resource and, if the specified resource name does not already begin with the value of *stype*, it is prefixed with the value of *stype*, separated by a period.

The *stype* values SUBSYSTEM and DEFAULTS are silently ignored.

This parameter is mutually exclusive with the resource values DEFAULTS and SUBSYSTEM and with the *request_type* values CFGINFO, CFGONLY, CFG-ONLY, STATUS and STATUS-ONLY. Data retrieval does not occur for any other combination of the *request_type* and *stype* parameters.

TGPFX=variable_prefix

Specifies the variable prefix that is used to create the names of the task global variables that AOCQRY provides the retrieved information in.

The value of *variable_prefix* must be 3 characters long and defaults to SUB.

If you call AOCQRY from a routine that is driven from the automation control file you must specify a variable prefix for TGPFX= that is something other than SUB or you will corrupt the task global variables that are used by the routine that is driving your routine. This can lead to unpredictable behavior.

PARENTINFO

Specifies whether parent task global variable information is retrieved. Specifies whether information about the parent of the resource is retrieved and provided in task global variables.

The following values can be specified:

YES

Parent information is retrieved and provided in task global variables. If the dependency has a sequence number, PARENTINFO defaults to YES. Otherwise, PARENTINFO=NO applies.

Note: If the subsystem has multiple dependencies with sequence numbering, information is obtained for the first parent resource in the parent list that is recognized by the local System Automation Agent. If information is required for all the supporting resources, AOCQRY must be issued for each of them. A list of the supporting resources can be obtained from the SUBSPARENT task global variable.

NO Parent information is not obtained.

This value is enforced for *request_type* values CFGONLY, CFG-ONLY and STATUS-ONLY.

EXITS

This parameter determines how automation flag exits are invoked. The following values can be specified:

FORCED

When FORCED is specified, automation flag exits are invoked during flag evaluation regardless of the automation flag setting.

BYPASS

When BYPASS is specified, defined exits to an automation flag of a resource are executed, when this automation flag is checked during flag evaluation and when the flag value is E.

This is the default value.

Restrictions and Limitations

AOCQRY does not clear its task global variables before resetting them. Thus these task global variable may contain data from an earlier AOCQRY call that was for a different subsystem.

Return Codes

- 0 The function completed successfully. If checking an automation flag, automation is allowed.
- 1 The global automation flag is off.
- 2 The specific automation flag is turned off.
- 3 A valid resource was not found in the automation control file. This is not used if the *stype* parameter is coded.
- 4 Incorrect parameters were used in the call.
- 6 SA z/OS initialization is incomplete. Unable to process command request.

Note: If AOCQRY processes with a return code greater than 2, no task global variables are updated.

Usage

- AOCQRY accesses the automation flag settings as they are defined via the customization dialog or dynamically changed via the INGAUTO command to determine whether automation should continue.
- Return codes 1 and 2, which specify that automation is turned off, are set when automation flags are set to NO or are disabled for a certain time.
- Return code 3 indicates that this application is not defined to SA z/OS and therefore no automation should be done for it.
- The AOCQRY command searches through the automation flags in a predefined sequence to decide whether automation should continue. The first Automation flag entry defined in the automation policy governs whether automation is allowed. The search order is:
 1. The flags that are associated with the fully-qualified resource name as derived from the input parameters.
 2. If the resource name consists of several qualifiers, the flags associated with the resource name, which has been truncated by one qualifier.
 3. If the last remaining resource qualifier is the name of a subsystem, the flags associated with SUBSYSTEM.
 4. The flags associated with DEFAULTS.
- If the *request_type* is coded as Initstart, Start, Recovery, Terminate or Restart, a two-level search is performed. The predefined search sequence described above is performed twice: first for the Automation flag, and then for the specific automation flag. If the Automation flag turns off automation, the second search is not performed and AOCQRY processing terminates.

Task Global Variables

There are three main groups of task global variables that are provided by AOCQRY:

- Application information task global variables (SUBSxxxxx)
- Parent information task global variables (SUBPxxxxx)
- Automation flag task global variables

If *stype* is not coded in the AOCQRY call, the following apply:

- All task global variables are modified unless parent information is not requested or parent information is not valid.
- If an application entry is not found, the task global variables are not altered from previous settings.

If *stype* is coded in the AOCQRY call, only SUBSAPPL, SUBSTYPE and AUTOTYPE are modified. All other task global variables retain their previous value.

Table 4 on page 45 lists AOCQRY application task global variables (SUBSxxxxx task global variables).

Table 4. AOCQRY Subsystem Task Global Variables

Task Global Variable	Description
SUBSAPPL	The application name from the automation control file. If <i>stype</i> was coded, this task global variable contains the resource name.
SUBSASID	The address space ID of the application. This is only available when SA z/OS process monitoring is used for this resource.
SUBSCATEGORY	The subsystem category (for example, JES2, DB2®, IMS™, USS, etc.).
SUBSCMDPFX	The application command prefix from the automation control file.
SUBSDDESC	The application description from the automation control file.
SUBSEXTSTART	Contains the 'External Start' information.
SUBSEXTSTOP	Contains the 'External Stop' information.
SUBSFILE	Contains the information whose file this resource represents.
SUBSFILTER	Contains the filter information that is associated with the path specification of the USS resource.
SUBSINFOLINK	The application INFOLINK from the automation control file.
SUBSIPOPT	The application IPL option from the automation control file.
SUBSIPSTACK	Contains the name of the IP stack to be used for port monitoring of USS resources.
SUBSJOB	The application job name from the automation control file.
SUBSJOBTYPE	The subsystem jobtype from the automation control file (MVS/NONMVS/TRANSIENT).
SUBSMDATE	The date the last monitor cycle checked the subsystem.
SUBSMTIME	The time the last monitor cycle checked the subsystem.
SUBSOPER	The work operator assigned to the subsystem.
SUBSPARENT	A list of the application parent names from the automation control file. The parent names are separated by blanks. This is only provided if dependencies with a sequence number were specified in the dialog.
SUBSPATH	Contains the information whose z/OS UNIX process this resource represents.
SUBSPID	The ID for the USS process.
SUBSPLEX	The name of the plex that is associated with the subsystem.
SUBSPORT	Contains the information whose TCP port this resource represents.
SUBSPROC	Contains the subsystem's PROCNAME.
SUBSPROCESS	Contains the current process (START, STOP, or null).
SUBSRSTOPT	The application restart information from the automation control file.
SUBSSCHEDSS	The application scheduling subsystem from the automation control file. If not specified, it defaults to the primary scheduling subsystem.
SUBSSDATE	The date the status of the subsystem was last updated.
SUBSSESS	The subsystem name from the automation control file.
SUBSSHUTDLY	The application shutdown delay value from the automation control file.
SUBSSPARM	The application parameter data from the automation control file.
SUBSSTARTTYPE	The application start type. This value is only available if the application is currently in a start phase.
SUBSSTAT	The application status from the automation status file.
SUBSSTIME	The time the status of the subsystem was last updated.
SUBSSTOPTYPE	The application stop type. This value is only available if the application is currently in a stop phase.

AOCQRY

Table 4. AOCQRY Subsystem Task Global Variables (continued)

Task Global Variable	Description
SUBSSTRTCYC	The application start cycles from the automation control file.
SUBSSTRTDLY	The application start delay from the automation control file.
SUBSSUBCAT	The subsystem subcategory (for example, tracker, TOR, AOR, etc.).
SUBSSUBID	The subsystem ID of the application as defined in the customization dialog.
SUBSSUBTYPE	The subsystem type (JES2, JES3, DB2, CICS, or IMS). Note: This task global variable is obsolete and is provided only for compatibility. Use SUBSCATEGORY instead.
SUBSSYMBOL _n	The application symbol defined for the subsystem ($n=1-9$).
SUBSTERMDLY	The application termination delay from the automation control file.
SUBSTRANTY	Used by transient subsystems to indicate whether they can be rerun.
SUBSTYPE	This task global variable indicates the resource that automation flag checking is performed for. For an application, the value for this task global variable is SUBSYSTEM. For resources other than applications, the value for this task global variable is the value coded for <i>stype</i> on the AOCQRY call. If an application entry was not found, the task global variable value is NONE.
SUBSUSER	Contains the information whose z/OS UNIX user ID this resource belongs to.
SUBSUSSJOB	The real job name of the application. This is only available when SA z/OS process monitoring is used for this resource.
SUBSWLMNAME	A list of the workload manager names from the automation control file.
SUBSWTOR	The list of reply IDs of primary outstanding WTORs of the application.

Table 5 lists AOCQRY parent task global variables (SUBPxxxx task global variables).

Table 5. AOCQRY Parent Task Global Variables

Task Global Variable	Description
SUBPAPPL	The parent application name.
SUBPASID	The parent address space ID. This is only available when SA z/OS process monitoring is used for this resource.
SUBPCATEGORY	The parent subsystem category (for example, JES2, DB2, IMS, USS, etc.).
SUBPCMDPFX	The parent command prefix from the automation control file.
SUBPDESC	The parent description
SUBPEXTSTART	Contains the 'External Start' information.
SUBPEXTSTOP	Contains the 'External Stop' information.
SUBPFILE	Contains the information whose file the parent represents.
SUBPFILTER	Contains the filter information that is associated with the path specification of the USS resource.
SUBPINFOLINK	The parent INFOLINK from the automation control file.
SUBPIPSTACK	Contains the name of the IP stack of the parent subsystem.
SUBPIPLOPT	The parent IPL option from the automation control file.
SUBPJOB	The parent job name.
SUBPJOBTYPE	The parent subsystem jobtype from the automation control file (MVS/NONMVS/TRANSIENT).
SUBPMDATE	The date the last monitor cycle checked the parent subsystem.

Table 5. AOCQRY Parent Task Global Variables (continued)

Task Global Variable	Description
SUBPMTIME	The time the last monitor cycle checked the parent subsystem.
SUBPOPER	The work operator assigned to the parent.
SUBPPARENT	A list of the parent names from the automation control file. The parent names are separated by blanks. This is only provided if dependencies with a sequence number were specified in the dialog.
SUBPPATH	Contains the information whose z/OS UNIX process the parent represents.
SUBPPID	The ID for the USS process of the parent.
SUBPPLEX	The name of the plex that is associated with the subsystem.
SUBPPORT	Contains the information whose TCP port the parent represents.
SUBPPROC	Contains the subsystem's PROCNAME.
SUBPPROCESS	Contains the current process (START, STOP, or null).
SUBPRSTOPT	The parent restart information
SUBPSCHESS	The scheduling subsystem for the parent, from the automation control file. If not specified, it defaults to the primary scheduling parent.
SUBPSDATE	The date the status of the parent system was last updated.
SUBPSESS	The parent subsystem name from the automation control file.
SUBPSHUTDLY	The parent shutdown delay value
SUBPSPARM	The parent parameter data
SUBPSTAT	The parent status
SUBPSTARTTYPE	The application start type. This value is only available if the application is currently in a start phase.
SUBPSTIME	The time the status of the parent system was last updated.
SUBPSTOPTYPE	The application stop type. This value is only available if the application is currently in a stop phase.
SUBPSTRTCYC	The parent start cycles from the automation control file.
SUBPSTRTDLY	The parent start delay from the automation control file.
SUBPSUBCAT	The parent subsystem subcategory (for example, tracker, TOR, AOR, etc.).
SUBPSUBID	The subsystem ID of the application as defined in the customization dialog.
SUBPSUBTYPE	The subsystem type of the parent subsystem.
SUBPSYMBOL _n	The application symbol defined for the parent ($n=1-9$).
SUBPTERMDLY	The parent termination delay from the automation control file.
SUBPTRANTY	If the parent subsystem is a transient, this indicates whether it can be rerun.
SUBPTYPE	This task global variable indicates the resource that automation flag checking is performed for. For an application, the value for this task global variable is SUBSYSTEM. For resources other than applications, the value for this task global variable is the value coded for <i>stype</i> on the AOCQRY call. If an application entry was not found, the task global variable value is NONE.
SUBPUSER	Contains the information whose z/OS UNIX user ID the parent belongs to.
SUBPUSSJOB	The real parent job name. This is only available when SA z/OS process monitoring is used for this resource.
SUBPWLMNAME	A list of the workload manager names from the automation control file.
SUBPWTOR	A list of reply IDs of primary outstanding WTORs of the application.

AOCQRY

Note: The SUBP variables are only available if a relationship with a sequence number was specified in the customization dialog.

Table 6 lists AOCQRY automation flag task global variables.

Table 6. AOCQRY Automation Flag Task Global Variables

Task Global Variable	Description																		
AUTOTYPE	<p>The AUTOTYPE task global variable contains the value of the automation mode that is turned off. Depending on certain conditions, AUTOTYPE has the following values:</p> <table><thead><tr><th>Value</th><th>Condition</th></tr></thead><tbody><tr><td>Null</td><td>Automation is allowed.</td></tr><tr><td>Null</td><td><i>request_type</i> does not check automation flags.</td></tr><tr><td>GLOBAL</td><td>The Automation (global) automation flag is off.</td></tr><tr><td>INITSTART</td><td>Initstart automation flag is off.</td></tr><tr><td>RECOVERY</td><td>Recovery automation flag is off.</td></tr><tr><td>RESTART</td><td>Restart automation flag is off.</td></tr><tr><td>START</td><td>Start automation flag is off.</td></tr><tr><td>TERMINATE</td><td>Shutdown automation flag is off.</td></tr></tbody></table>	Value	Condition	Null	Automation is allowed.	Null	<i>request_type</i> does not check automation flags.	GLOBAL	The Automation (global) automation flag is off.	INITSTART	Initstart automation flag is off.	RECOVERY	Recovery automation flag is off.	RESTART	Restart automation flag is off.	START	Start automation flag is off.	TERMINATE	Shutdown automation flag is off.
Value	Condition																		
Null	Automation is allowed.																		
Null	<i>request_type</i> does not check automation flags.																		
GLOBAL	The Automation (global) automation flag is off.																		
INITSTART	Initstart automation flag is off.																		
RECOVERY	Recovery automation flag is off.																		
RESTART	Restart automation flag is off.																		
START	Start automation flag is off.																		
TERMINATE	Shutdown automation flag is off.																		
EHKEXITRSN	The return code from the exit if a nonzero return code.																		
EHKEXITNME	The name of the exit supplying the nonzero return code.																		
SUBSASSIST	The Assist Mode setting for the automation flag.																		

Examples

Example 1

This example shows how AOCQRY can be used in an automation procedure to check whether automation is allowed for an application, before invoking automation actions.

Assume that the CICST subsystem issues a message during termination that invokes this automation procedure via a NetView automation table statement. Assume also that the message identifier is not considered for the automation decision.

The automation procedure to call AOCQRY is:

```
/* REXX CLIST to check if termination automation is allowed for CICST */
'AOCQRY CICST,TERMINATE'
Select
  When rc = 0 Then Do          /* Automation on; perform actions req'd. */
    :
  End
  When rc = 1 | rc = 2 Then Do /* Automation off; If applicable log a
    :                          message indicating unable to take
    :                          action for message. */
  End
  When rc = 3 Then Exit        /* Subsystem not automated */
  Otherwise Do                 /* Error; log error message */
    :
  End
End
Exit
```

The automation procedure performs the following processing steps:

1. The automation procedure calls AOCQRY, supplying CICST as the subsystem name and TERMINATE as the *request_type*. AOCQRY retrieves the appropriate subsystem information, and then searches for the automation flags.
2. After returning from AOCQRY, the automation procedure determines what the return code was, then takes the appropriate action.

Supposing that the subsystem CICST has a HasParent relationship to VTAM®, AOCQRY accesses both the definitions of CICST and VTAM to fill in the task global variables. All the SUBSxxxx task global variables are filled with the CICST subsystem information, and the SUBPxxxx task global variables are filled with the VTAM information.

Example 2

The automation procedure from Example 1 can be coded in a more generic way by using the NetView REXX function jobname() as the resource option of AOCQRY.

The automation procedure to call AOCQRY is:

```

/* REXX CLIST to check if termination automation is allowed for a job
- generic check dependant on Jobname                                     */
'AOCQRY 'Jobname()',TERMINATE'
Select
  When rc = 0 Then Do          /* Automation on; perform actions req'd.  */
    :
    End
  When rc = 1 | rc = 2 Then Do /* Automation off; If applicable log a
                                message indicating unable to take
                                action for message.                       */
    :
    End
  When rc = 3 Then Exit      /* Subsystem not automated      */
  Otherwise Do              /* Error; log error message     */
    :
    End
End
Exit

```

The jobname() function returns the name of the job that has issued the message. Using this function, the automation procedure supports any job that can issue the message that invokes this automation procedure via the NetView automation table. This allows portability of the automation procedure to different systems without requiring changes to it. The job name is supplied only to an automation procedure that is called from the NetView automation table.

If your automation procedure issues MSGREAD commands, you must issue the jobname() function upon entry because the returned value resets whenever the MSGREAD command is issued.

Example 3

This example shows how AOCQRY can be used in an automation procedure to check whether automation is allowed for an MVS component, before invoking automation actions.

The message to automate is issued by MVS indicating that an SMF dump data set is full. The automation procedure verifies that automation is allowed by calling AOCQRY.

The automation procedure to call AOCQRY is:

AOCQRY

```
/* REXX CLIST to check if recovery automation is allowed for SMF */
'GLOBALV GETC AOFSYSTEM'
'AOCQRY SMFDUMP,RECOVERY,'aofsystem
Select
  When rc = 0 Then Do /* Automation on; perform actions req'd. */
  :
  End
  When rc = 1 | rc = 2 Then Do /* Automation off; If applicable log a
  message indicating unable to take
  action for message. */
  :
  End
  Otherwise Do /* Error; log error message */
  :
  End
End
Exit
```

This example differs from the previous examples as follows:

- When automating an MVS component, use a generic component as the resource name when calling AOCQRY instead of a message identifier. Thus several messages can be related to a single MVS component to be automated.
- The third parameter *stype* is coded. Coding *stype* tells AOCQRY to skip the process of finding subsystem entries. The example uses the AOFSYSTEM common global variable as the *stype* parameter. The value of the variable is MVSESA.
- Return Code 3 is not valid because the application entries in the automation control file are not checked.

AOCUPDT

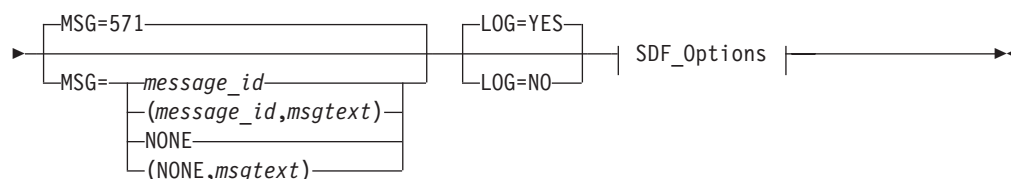
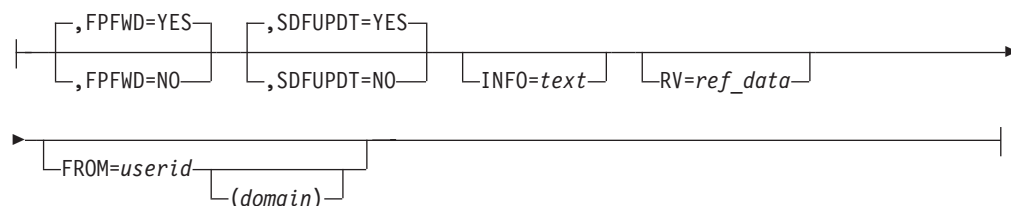
Purpose

AOCUPDT performs several status update functions, including:

- Updating the automation agent status for the resource
 - Identifying any messages associated with the automation agent status change and processing options performed for these messages:
 - Whether a message is issued and logged in the NetView log
 - Which message is issued
 - Whether the message is sent as a notification message to notification operators on a local system (distinct from forwarding to a focal-point system)
 - Whether the message is forwarded to a focal-point system
- AOCUPDT calls the AOCMSG command to handle processing of log and notification messages.
- Updating SDF status displays with the resource status change
 - Updating the automation manager OBSERVED status

Syntax

►►—AOCUPDT—*resource*—STATUS=*status*—RESTYPE=SUBSYSTEM
RESTYPE=*type*—►

**SDF_Options:****Parameters***resource*

The name of the resource that status or information updates specified by other AOCUPDT parameters are performed for. This value is required and must be specified first on an AOCUPDT call. You can use the following formats for *resource*.

Format	Example
<i>system_name.resource</i>	PROD.TSO or PROD.VTAM
<i>resource</i>	TSO or VTAM

The *system_name* variable defaults to AOFSYSNAME.

STATUS

Specifies the new value of the automation agent status for the resource.

When you use this parameter to change status, some other AOCUPDT parameters perform default actions, unless otherwise coded. These parameters include:

- MSG
- FPFWD
- SDFUPDT

If you specify STATUS to change status, but do not specify any of the parameters listed above (thereby using parameter defaults), the following occurs:

- SA z/OS issues the message A0F571I *resource_name* SUBSYSTEM STATUS FOR JOB *jobname* IS *status - text* and also logs the message in the NetView log.
- The specified status change is reflected in SDF status panels.

To change the values or actions performed by the MSG, FPFWD, and SDFUPDT parameters, or to preclude their use, you must specify those parameters and desired values.

If a status value that has a length greater than eight characters is used, the status value is truncated to a length of eight characters.

RESTYPE

Identifies the type of resource for the *resource* parameter. You can specify a resource type of your own choice, with the exception of SYSTEM, which is reserved for internal use only. The default is SUBSYSTEM.

type

A resource type of your own choice.

SUBSYSTEM

A resource type of SUBSYSTEM.

FPFWD

Determines whether the specified status is sent from a local system (the system that AOCUPDT is issued on) to a focal-point system.

YES

The status is sent. This is the default.

NO The status is not sent.

Note: For status forwarding to a focal-point system to occur, you must already have configured an automation network and defined the automation network to SA z/OS. Refer to *IBM Tivoli System Automation for z/OS User's Guide* for details.

SDFUPDT

Determines whether the specified status change is also reflected in SDF status displays.

YES

The status change is reflected in SDF status displays. This is the default if STATUS is specified.

NO The status change is not reflected in SDF status displays. This is the default if STATUS is not specified.

MSG

This parameter identifies the message associated with the status change specified with the STATUS parameter. This message is issued to make a note of when the status change occurs. This parameter is applicable only if the STATUS parameter is also specified.

The default is 571, the message ID for SA z/OS status change message AOF571I, *resource_name* SUBSYSTEM STATUS FOR JOB *jobname* IS *status—text*.

This parameter value can be specified using the following formats:

message_id

Identifies the numeric part of a message ID. For example, 571 specifies SA z/OS message AOF571I.

(message_id)

The complete message ID, including message prefix and message number, enclosed in parentheses, for example, (AOF123).

(message_id,msgtext)

The complete message ID, including message prefix and message number, plus message text to be substituted for message variables in the message text. This entire specification is enclosed in parentheses, for example, (123,AA,BB,CC). Quotation marks are not allowed in the message text.

The AOCMSG command substitutes the message text values into message variables &1 through &9 in the fixed message text that is located in the NetView message library. See "AOCMSG" on page 34 for details of how

this command works. Some message variables are preset to certain values depending on the message ID and message text that are specified on the AOCUPDT call:

- Variable &1 is always set to AOFRUPDT, which is the name of the automation procedure that the AOCUPDT command processor resides in.
- If the message text is omitted, the following message variables are preset:

Var	Setting
&1	AOFRUPDT
&2	Time
&3	<i>system_name.resource</i>
&4	Resource type
&5	Subsystem name
&6	Subsystem job name
&7	Status

- If the message text is provided and the *message_id* number is 571, the following message variables are preset:

Var	Setting
&1	AOFRUPDT
&2	Time
&3	<i>system_name.resource</i>
&4	Resource type
&5	Subsystem name
&6	Subsystem job name
&7	Status

Variables &8 and &9 can be assigned values from the *msgtext* portion of this parameter.

- If the message text is provided and the *message_id* number is not 571, the following message variables are preset:

Var	Setting
&1	AOFRUPDT
&2	Time
&3	<i>system_name.resource</i>
&4	Resource type

Variables &5 through &9 can be assigned values from the *msgtext* portion of this parameter.

NONE

The operator is not notified that the update has taken place. The text string "RESOURCE *resource_name* STATUS UPDATED TO *status_value*" is written to SDF.

(NONE,*msgtext*)

The operator is not notified that the update has taken place. The text string "*msgtext*" is written to SDF.

LOG

Specifies whether the trace message is written in the NetLog. The default is YES. The message has the following format: AOFRUPDT: *system_name* *resource_name* *resource_type*,STATUS=*status*

INFO

The SDF INFO attribute shown in the detail panel of an SDF status descriptor. Can be any alphanumeric character up to a length of 80 characters.

AOCUPDT

| RV The reference value of the SDF status descriptor. The maximum length is 40
| characters.
|
| FROM
| The user id (operator task) and optionally the domain id of the originator of
| the status update. The default is the issuing task and local domain id.

Restrictions and Limitations

AOCUPDT has the following restrictions and limitations:

- AOCUPDT should only be issued from an automation procedure.
- Parentheses appearing within message text must be properly paired and balanced.
- Using AOCUPDT to change a resource status *only* changes the status. It does not initiate any associated status change processing that occurs if the status change is processed through a command such as ACTIVMSG or TERMMSG. Also, the automation status remains unchanged. For example, if the resource is involved in a STARTUP, and the resource's status is changed to UP via AOCUPDT, this process will not be affected because the automation status will not be changed to IDLE.

Return Codes

- 0 AOCUPDT processed normally.
- 4 All requested actions were performed. However, the system detected that some of the data to be changed was the same as the modified data specified on the AOCUPDT call.
- 8 Incorrect keyword specifications were detected and ignored. All other keywords processed normally.
- 12 No function keyword was specified on the AOCUPDT call. A resource was identified, but no action to perform on the resource was specified.
- 16 The specified *resource* was not found, when the specified resource type (RESTYPE value) is SUBSYSTEM and the system name is the system that AOCUPDT is running on.
- 20 The *resource* name length was longer than allowed. When the specified RESTYPE value is SUBSYSTEM, the resource name cannot be longer than 11 characters.
- 99 A timeout or another error occurred.

Usage

- When you use AOCUPDT to change resource status, the status change message is sent to notification operators defined to receive the message. Notification of a status change occurs whether automation flags for the resource are enabled or disabled (set to Yes or No). To suppress sending a message when a status change occurs, specify MSG=NONE along with the STATUS parameter.

Examples

Example 1

This example shows how to use AOCUPDT to change the status of the TSO subsystem to UP.

```
AOCUPDT TSO,STATUS=UP
```

Note: This will not cause any TSO UP commands to be issued.

Example 2

This example shows how to use AOCUPDT to:

- Change the status of subsystem IMSPROD to DOWN
- Issue a customer-defined message, ABC123
- Ensure that the status change is *not* reflected in SDF

```
AOCUPDT IMSPROD,STATUS=DOWN,MSG=ABC123,SDFUPDT=NO
```

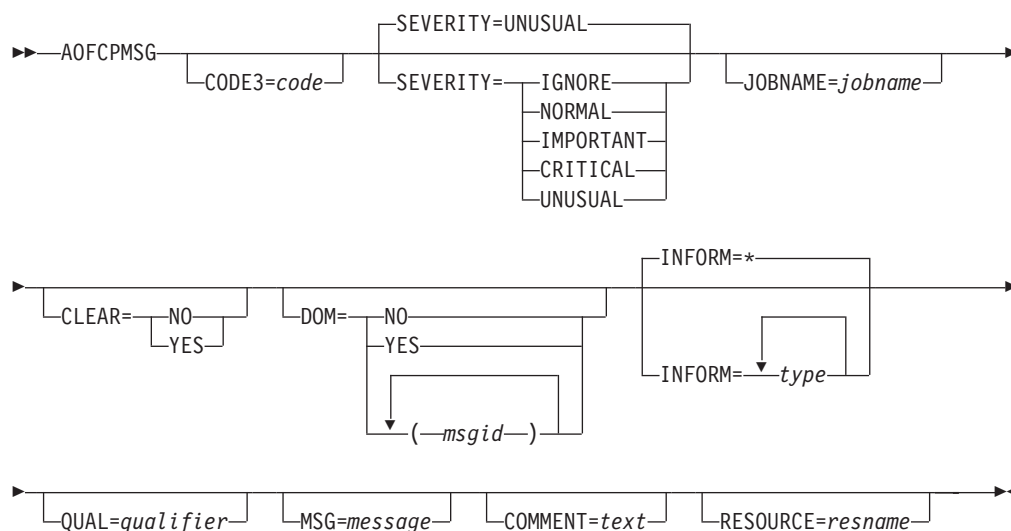
AOFCPMSG

Purpose

The AOFCPMSG command lets you:

- Capture messages and save them in common global variables for subsequent display by DISPINFO
- Add the message to SDF for display in the Messages panels
- Add the message to NMC as a minor resource of the major resource that issued the message

Syntax



Parameters

CODE3

code

This is the optional CODE3 value used by CDEMATCH to specify the severity of the message.

SEVERITY

This parameter allows you to directly specify a severity and bypass the code matching process. To change the severity classification of a message you need to change the NetView automation table (AT).

The severity is always overwritten by a policy entry. That is, if a policy entry exists, the severity is taken from there.

The severity of a message can also be specified in a CDEMATCH against the subsystem. If no match is found against the subsystem, a match is attempted against the system issuing the message. The message ID for the code match is CAPMSGs.

CODE1 is set to the message ID of the message being captured. CODE2 is set to the job name (you can alternatively set this to the subsystem name) of the subsystem that issued the message. CODE3 is set to the value specified in the *code* parameter.

The severity code that is defined as the value to be returned to the command list can be one of the following:

IMPORTANT	The message is captured and its color is set to PINK.
IGNORE	The message is not captured.
NORMAL	The message is captured and its color is set to GREEN, but not forwarded to SDF or NMC.
UNUSUAL	The message is captured and its color is set to YELLOW.
CRITICAL	The message is captured and its color is set to RED.

JOBNAME

The JOBNAME is optional and specifies the job name of the subsystem that issued or is assigned to the message. This parameter overrides the value as determined from the jobname() function against the message.

CLEAR

Specifies whether the existing messages that are recorded for the subsystem should be erased and SDF and NMC resources should be removed for the subsystem. This may be specified without a message being issued. The default is NO.

DOM

Specify YES to cause AOFCPMSG to delete a previously captured message instance that matches the current message ID (that is, the message ID that is passed to AOFCPMSG from the AT, unless it is overridden by the MSG parameter).

Note: This will not capture the latest copy of the message being deleted. See "The MSG Parameter" on page 57 for details of how to override the message ID that is passed from the AT.

You can also specify one or more IDs of messages to be deleted. This allows you to capture a message and thereby delete other messages that become obsolete in one step. The message IDs must be separated by a blank character and enclosed in parenthesis or quotes if more than one message ID is specified. The message ID can contain wildcards.

The default is NO unless the message is detected as a DOM, in which case the default is YES.

Note: Messages that have been DOMed by an operator are still displayed on SDF, NMC and TEP. To remove the message explicitly, you must use the INGMSGs command or its equivalent in SDF, NMC and TEP. See INGMSGs in *IBM Tivoli System Automation for z/OS Operator's Commands*.

INFORM

Specifies the target destination where the message should be sent to. If omitted, the inform list definition in the automation policy that is associated with the resource is used.

QUAL

Specifies a qualifier that is used to identify the message in the absence of the message arrival time (MAT) when deleting the message. The qualifier consists of two elements, separated by a slash (/):

1. The first element is the subcomponent that is associated with the resource, for example, CICSSOS for a CICS short on storage condition.
2. The second element identifies the message uniquely within the subcomponent and resource. This parameter is optional.

MSG

Specify a message to override the message that is passed from the AT. This may be an entire message including a message ID followed by message text or just a message ID. This parameter can also be used in conjunction with the DOM parameter to DOM any previously captured message. When used in conjunction with the DOM parameter only a message ID is needed.

If *message* contains blanks or special characters it must be delimited with single quotation marks, double quotation marks, or parentheses.

COMMENT

If specified, this text will be appended to the message for SDF and placed in the DATA3 field for NMC.

The comment text must be delimited with single quotation marks, double quotation marks, or parentheses if it contains blanks or special characters.

RESOURCE

The name of the resource that is to be associated with the message. This is typically specified in automation manager notation but can also be the subsystem name or any other name.

The RESOURCE parameter takes precedence over the JOBNAME parameter if it is also specified.

Restrictions and Limitations

To use the AOFCPMSG command SA z/OS must be initialized.

AOFCPMSG should only be executed as a NetView automation table command.

Excessive use of AOFCPMSG will reduce NetView storage because messages are stored in common global variables.

It is recommended that you restrict the use of AOFCPMSG to exception condition messages.

Return Codes

- 0 The AOFCPMSG command completed successfully.
- 1 Incorrect parameters were used.
- 2 The job name could not be resolved to a subsystem and is not MVSESA. System messages are only cleared if the job name is specified as MVSESA.
- 3 The AOFCPMSG command was called without a message in its default safe. This is only valid if CLEAR=YES is specified.
- 6 Automation is not initialized.

Usage

You should add the AOFCPMSG command to your NetView automation table (AT). However, you can also call AOFCPMSG outside of the AT when specifying the MSG parameter.

The severity of the message can be specified in a CDEMATCH against the subsystem or, if not found there, against the system that the subsystem or resource belongs to. The following settings apply:

CODE1

The message ID.

CODE2

The job name of the subsystem.

Note: In earlier releases, this was the subsystem name.

CODE3

User-defined (optional).

Value Returned

The severity (message ID p₁ p₂ ... p_n).

Optionally, a message ID, followed by one or more message parameters, can be specified. If present, the resulting text is appended to the message for SDF and placed in the DATA3 field for NMC.

Examples

The following example shows how AOFCPMSG is called from the NetView automation table to change the severity of message HSAM1050E to CRITICAL:

```
IF MSGID = 'HSAM1050E'
THEN
EXEC(CMD('AOFCPMSG SEVERITY=CRITICAL')ROUTE(ONE * %AOFOPGSSOPER%));
```

AOFEXCMD

Purpose

AOFEXCMD is used to execute a command on a specified autotask. If the autotask is not active, AOFEXCMD will try to execute the command on a backup autotask. This process is repeated until the command is successfully scheduled for execution, or the list of available backup autotasks is exhausted.

AOFEXCMD will attempt to execute the command on the following autotasks:

1. The primary autotask for the automated function, *autofunc*
2. The secondary autotask for the automated function, *autofunc*
3. The primary autotask for SYSOPER
4. The secondary autotask for SYSOPER
5. The primary autotask for BASEOPER
6. The primary autotask for AUTO1

If the command cannot execute on any of these autotasks, an AOF572I or AOF763I message is issued.

SA z/OS automation will attempt to restart any inactive autotasks that are called by AOFEXCMD.

Syntax

▶▶ AOFEXCMD *autofunc,command* ▶▶

Parameters

autofunc

The automated function that the autotask name is defined under. Automated functions are established in the customization dialogs, and are assigned at SA z/OS initialization.

If the specified value is not defined as an automated function, it is considered to be an operator ID or task name.

If the automated function name is not supplied the procedure will attempt to issue the command on one of the backup automated functions (SYSOPER, BASEOPER or AUTO1).

Note: If the automated function name is not supplied, a comma must be used as a placeholder for parsing so that the command is identifiable as the second operand.

command

The command that is to be scheduled to run on the autotask associated with the automated function.

Restrictions and Limitations

If the command cannot be routed to one of the following automated functions, command execution is not attempted on a backup autotask:

- GATOPER
- GEOxxx
- PLEXOPR2
- PLEXOPR3

Issuing operators must be authorized to issue the command using EXCMD.

Messages

The following message is issued by AOFEXCMD when it has failed to execute on any of the available autotasks. This may occur if AOFEXCMD is issued before SA z/OS initialization is complete.

```
AOF572I CGLOBALS NOT INITIALIZED FOR AUTOMATED FUNCTION autofunc -
UNABLE TO ROUTE COMMAND command, operand_1, operand_2, operand_3
```

Example

The following command schedules a message to be sent from autotask AUTNET1 to operator OPER1. In this example, AUTNET1 is defined under the automated function NETOPER.

```
AOFEXCMD NETOPER,MSG OPER1 Logoff in 5 mins
```

AOFRACON

Purpose

Use the AOFRACON command to assign an autotask to each MCS console that is not already served by a NetView operator.

If the console *ANY* is assigned to a NetView operator, AOFRACON does not perform console assignments and terminates with RC=1.

If the console *MASTER* is assigned to a NetView operator, the master console gets an autotask assigned anyway, because a switch of the master console would result in this console not having access to NetView.

Syntax

▶▶ AOFRACON ◀◀

Parameters

None.

Return Codes

- 0 Processing successful.
- 1 Processing not required (*ANY* assignment).
- 4 Processing failed.

Restrictions and Limitations

SA z/OS must be fully initialized prior to running AOFRACON.

A suitable exit to run AOFRACON from would be AOFEXINT.

AOFRMTR

Purpose

The AOFRMTR command can be used to update the health status of monitor resources by issuing the monitor commands that have been defined for these monitor resources.

The monitor commands that are issued for those monitor resources are those that are defined for a given object and job name.

Syntax

▶▶ AOFRMTR *object* _jobname_ ◀◀

Parameters

object Specifies the object that the monitor resource is associated with.

jobname

Specifies the job name that the monitor resource is associated with. If AOFRMTR is called from the NetView automation table, the issuing job of the automated message is taken as the default value.

Usage

You should normally call the AOFRMTR command from the NetView automation table.

AOFSET

Purpose

The AOFSET command is used to set the agent resource attributes to ABENDING or BREAKING for a given subsystem on the specified system.

Syntax

▶▶—AOFSET—*system*—*subsystem*—*function*—▶▶

Parameters

system

The system that the agent resource parameters are to be changed on.

subsystem

The subsystem that the agent resource parameters are to be changed for.

function

The agent resource parameters that are to be changed. The following values can be specified:

ABENDING

Set the next stop of the subsystem to ABENDING.

BREAKING

Set the next stop of the subsystem to BREAKING.

Restrictions and Limitations

ABENDING and BREAKING can only be used for active subsystems.

Return Codes

- 0 OK.
- 1 An error occurred. The cause of the error is described in the error message.
- 6 Environment not initialized.

AOFTREE

Purpose

The AOFTREE command is used to extract information about an application and its dependent applications. The information returned for each application in the parent-child hierarchy is:

- Name
- Job name

AOFTREE

- Type of the resource (can be application group or subsystem)
- Position in the tree

The relationship of an application to its dependent applications can be illustrated using a tree structure as in Figure 2.

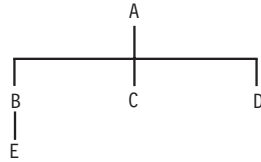
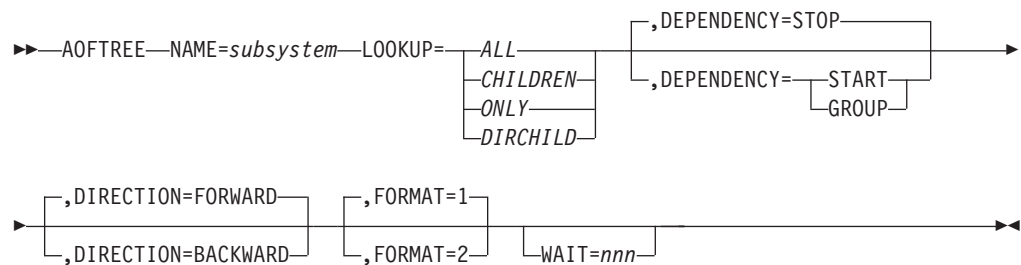


Figure 2. Subsystem Dependent Tree

Where:

- A, B, C, D, and E are all applications.
- A is the root of the tree. All applications below A are its dependants.
- B, C, and D are direct children of A, that is, dependent on A.
- B, C, D, and E are all dependants of A.
- A is the parent of B, C, and D. B is the parent of E.
- A is on level 1, B, C, and D are on level 2, and E is on level 3.
- E, C, and D have a position in the tree referred to as *LOWEST*. A and B have a position in the tree referred to as *UPPER*.

Syntax



Parameters

NAME

subsystem

The name of the application that you want to extract dependent applications for.

LOOKUP

The scope of the tree to be returned. The following values can be specified for lookup:

ALL

Returns details about the application and all its children.

CHILDREN

Returns details about all the children of the application.

ONLY

Returns details about the application only.

DIRCHILD

Returns details about the application and its direct children.

DEPENDENCY

Specifies the type of dependency (as defined in the Policy database) that the parent-child data should be returned for. The following options are available:

STOP

Returns all resources that are a child of the specified resource or that the resource has a stop dependency on. This is the default.

START

Returns all resources that are a parent of the specified resource or that the resource has a start dependency on.

GROUP

Returns all members that the specified resource consists of.

DIRECTION

Specifies the direction for returning the tree data. The following options are available:

FORWARD

This means progressing from the top level of the tree towards the bottom. This is the default.

BACKWARD

This means progressing from the bottom of the tree towards the top.

FORMAT

Specifies the output format that the information is returned in. The following options are available:

- 1 The data is returned in NetView task global variables (AOFPCCHILD.*n*). This is the default.
- 2 The data is returned in a multiline message.

WAIT

Specifies the number of seconds to wait before reporting that a timeout occurred if the automation manager does not provide the requested data. The maximum time interval is 999 seconds.

If omitted, the time interval is 30 seconds.

Restrictions and Limitations

This command can only be issued for a local system.

Return Codes

- 0 The command successfully executed. The results are in the task global variable array, AOFPCCHILD.
- 1 An invalid application name was used.
- 4 Invalid parameters were used.
- 5 Timeout or other error occurred.

Usage

When control is returned to the calling automation procedure, if the return code is not zero the AOFPCCHILD array is set to null.

Note: The AOFPCCHILD array is *not* sorted in hierarchical order.

AOFTREE

If you use the AOFTREE command within a PIPE and no parameters are passed, the contents of the default safe is taken and treated as input parameters. The output format is set to 2 (FORMAT=2).

Task Global Variables

AOFPCHILD.0

The number of elements in the array.

AOFPCHILD.n

The nth element in the AOFPCHILD array. This element contains the following details for a subsystem, separated by blanks:

- Name
- Job name
- Position in the tree

Examples

Figure 3 and the following examples show how the application dependency relationships are expressed using the AOFTREE command. The DIRECTION parameter setting is shown on the left hand side of the chart for an application tree structure.

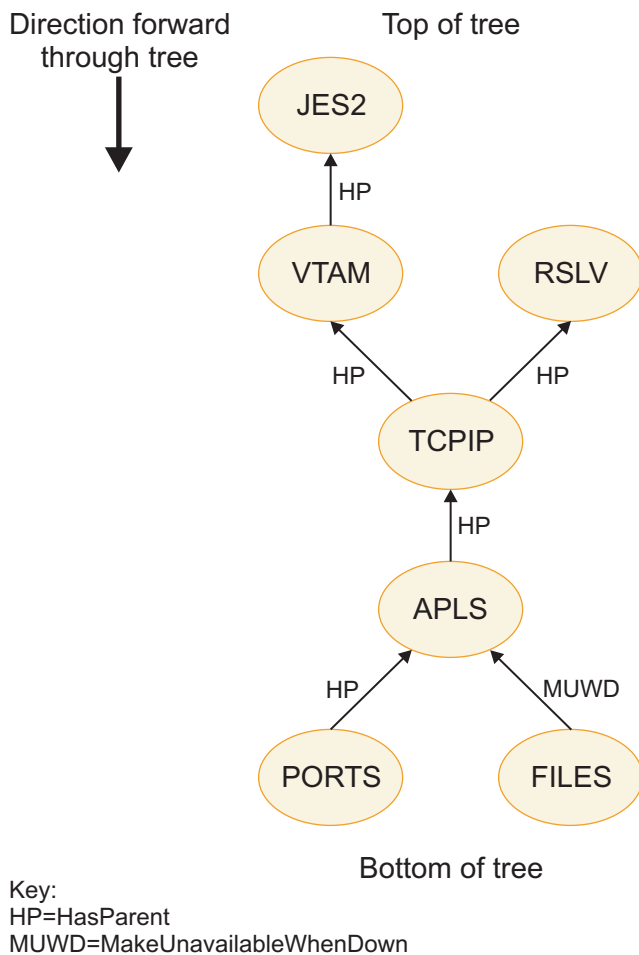


Figure 3. Application Tree

Example 1

Return all "children" from TCPIP using the STOP graph.

```
>>> AOFTREE NAME=TCPIP DEPENDENCY=STOP LOOKUP=ALL FORMAT=2
TCPIP/APL/AOC8 TCPIPS APL UPPER
APLS/APL/AOC8/ USSAPLS APL UPPER
PORTS/APL/AOC8 USSPORTS APL LOWEST
```

Example 2

Return all "parents" from TCPIP using the STOP graph.

```
>>> AOFTREE NAME=TCPIP DEPENDENCY=STOP LOOKUP=ALL DIRECTION=BACKWARD FORMAT=2
JES2/APL/AOC8 JES2 APL UPPER
RSLV/APL/AOC8 RESOLVRS APL UPPER
TCPIP/APL/AOC8 TCPIPS APL LOWEST
VTAM/APL/AOC8 VTAMS APL UPPER
```

Example 3

Return all STOP dependencies for FILES.

```
>>> AOFTREE NAME=FILES LOOKUP=ALL FORMAT=2
APLS/APL/AOC8 USSAPLS APL UPPER
FILES/APL/AOC8/ USSFILES APL LOWEST
PORTS/APL/AOC8 USSPORTS APL LOWEST
```

Example 4

Return all direct children from APLS using the STOP graph.

```
>>> AOFTREE LOOKUP=CHILDREN DIRECTION=FORWARD FORMAT=2
PORTS/APL/AOC8 USSPORTS APL LOWEST
```

The following results are obtained when calling AOFTREE with the following parameters, and the relationships between A, B, C, D, and E are as described in Figure 2 on page 62.

AOFTREE NAME=A, LOOKUP=ALL

```
AOFPCHILD.0 = 5
AOFPCHILD.1 = E Ejobname 0 LOWEST
AOFPCHILD.2 = B Bjobname 0 UPPER
AOFPCHILD.3 = C Cjobname 0 LOWEST
AOFPCHILD.4 = D Djobname 0 LOWEST
AOFPCHILD.5 = A Ajobname 0 UPPER
```

AOFTREE NAME=A, LOOKUP=ONLY

```
AOFPCHILD.0 = 1
AOFPCHILD.1 = A Ajobname 0 LOWEST
```

AOFTREE NAME=A, LOOKUP=DIRCHILD

```
AOFPCHILD.0 = 4
AOFPCHILD.1 = A Ajobname 0 UPPER
AOFPCHILD.2 = B Bjobname 0 UPPER
AOFPCHILD.3 = C Cjobname 0 LOWEST
AOFPCHILD.4 = D Djobname 0 LOWEST
```

AOFTREE NAME=B, LOOKUP=ALL

```
AOFPCHILD.0 = 2
AOFPCHILD.1 = E Ejobname 0 LOWEST
AOFPCHILD.2 = B Bjobname 0 UPPER
```

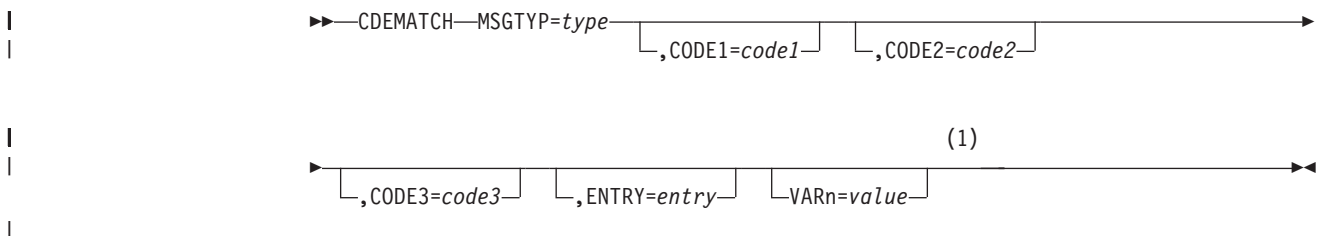
CDEMATCH

Purpose

The CDEMATCH command performs a function similar to a table search. It uses code values that are specified in the automation policy to create a table. You define the table match criteria and a control keyword or result field. Results from the search are returned to the automation procedure and are typically used to alter the automation procedure logic flow or an automation procedure command or reply.

A typical use is to extract feedback and return codes from the message you are automating, and then perform a search in the automation control file using those codes. The result of that search alters the action the automation procedure takes.

Syntax



Notes:

- 1 You can define VAR1 - VAR9.

Parameters

MSGTYP=*type*

This is the value that is entered in the **Message id** field for the MESSAGES/USER DATA policy item of the automation policy. This policy item is used to define the codes that are to be searched through for a match. MSGTYP is typically coded with the message ID of an automated message, but can also be a pseudo message ID such as CAPMSGs or INGALERT.

CODE1=*code1* CODE2=*code2* CODE3=*code3*

These code parameter values are used to search the code entries of the specified message ID for a matching code definition. You must supply at least one code parameter but you can supply all three code parameters, if desired. The code parameters can be specified in any order.

The values for the code parameters may have been extracted as variable values from an automated message but can also be any other values that a matching code definition is searched for. The code parameter values correspond to the related Code fields in the Code Processing panel of the MESSAGES/USER DATA policy item.

ENTRY=*entry*

This value is the entry:

- APL** The subsystem name
- MTR** The monitor name, prefixed with 0
- APG** The automation name, prefixed with 1
- MVS** MVSESA

It is the entry that can be viewed in the **Type** column in INGLIST.

The default value for this parameter is determined by the task global variables SUBSTYPE and SUBSAPPL. If the value of SUBSTYPE is SUBSYSTEM, the value of SUBSAPPL is taken as the default value for the ENTRY parameter. Otherwise, the value of SUBSTYPE is taken as the default value. You must call the AOCQRY command before CDEMATCH for the defaults to work.

VARn

Variables to substitute '&n' placeholders in the value returned. n can be 1 - 9.

Restrictions and Limitations

The CDEMATCH command can be called only by another automation procedure or a command processor.

Return Codes

- 0 A match was found. The resulting Value Returned of the matching code definition is provided in the task global variable EHKACTION.
- 1 No match was found.
- 4 Incorrect parameters were passed to the command.
- 6 SA z/OS initialization incomplete, unable to process command request.

Usage

- When a matching code definition is found in the automation policy for the passed parameters and control is returned to the calling automation procedure, task global variable EHKACTION contains the data from the Value Returned field in the Code Processing panel of the customization dialog. If no matching code definition is found, the value of the task global variable EHKACTION is null.
- Code matching specifications in the automation policy are order-dependent. The first matching code definition is used.
- Any code parameters (CODE1, CODE2, or CODE3) that are not specified when calling CDEMATCH are considered as matching regardless of what exists in the automation policy.
- The format of code matching specifications in the automation policy allows the use of the wildcard characters * and % and the use of comparison operators. For more details about the format of code definitions in the customization dialog see *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

Task Global Variables

EHKACTION

This variable provides the returned value of the matching code definition as specified in the Value Returned field in the Code Processing panel of the customization dialog.

When the return code for CDEMATCH is greater than zero, the value of this variable is null.

Example

This example shows the relationship between CDEMATCH and the automation control file. The message to automate, \$HASP095, is produced by the JES2 subsystem and indicates that a catastrophic-level problem has occurred. This example assumes the full message text is passed to the automation procedure. The automation procedure breaks the message apart, calls CDEMATCH to determine whether the error codes are in the automation control file.

The code matching information is specified in the automation policy as follows.

Select the MESSAGES/USER DATA policy item for the JES2 Application object. On the Message Processing panel for the JES2 subsystem enter COD as the action for the message \$HASP095.

The Code Processing panel for message \$HASP095 is displayed, as shown in Figure 4.

Code 1	Code 2	Code 3	Value Returned
*	\$PJ*		OPERCANCEL
ERROR*	\$K03		IPLREQ
ERROR*	\$K08		IPLREQ
ERROR*	\$K15		IPLREQ
ABEND*	SA22		OPERCANCEL

Figure 4. Code Processing Sample Panel

The automation procedure is as follows:

```

/* REXX CLIST to respond to $HASP095 */
/* Check whether automation is allowed and set TGLOBALs */
'AOCQRY JES2 RECOVERY'
:
/* Get text of triggering message and save it in msg.1 */
'PIPE SAFE * | STEM msg.1'

If msg.0 > 0 Then Do
  /* Parse the input message: */
  /* $HASP095 JES2 CATASTROPHIC type. CODE = cde RC=rc */
  Parse Var msg.1 'CATASTROPHIC' code1 . 'CODE =' code2 .
  /* Look for a match */
  'CDEMATCH MSGTYP=$HASP095, CODE1='code1', CODE2='code2'
  Select
    When rc = 0 Then Do /* Match found: check the action field */
      'GLOBALV GETT EHKACTION'
      If ehkaction = 'IPLREQ' Then Do
        :
      End
    End
    When rc = 1 Then Do /* No match found: warn if required */
      :
    End
    Otherwise /* Error: perform warning action */
      :
    End
  End
End

```

This is how the automation procedure proceeds:

1. The automation procedure gets the triggering message \$HASP095 from the PIPE default SAFE and stores the only message text line in stem variable *msg.1*.
2. The message text is parsed using literal string patterns to extract the error type and the error code in variables *code1* and *code2*. Both values are passed to CDEMATCH as parameters.
3. According to the code specifications in the automation policy, CDEMATCH first checks the passed error code (*code2*) for \$PJ in the first three characters. This checking traps both \$PJ2 and \$PJE, which indicate that command \$PJES2,ABEND or \$PJES2,ABEND,FORCE has been issued.
4. In the case of a match, CDEMATCH returns the value OPERCANCEL in the task global variable EHKACTION.

5. If no match occurs for the first code definition, CDEMATCH checks for the error codes \$K03, \$K08, or \$K15 and returns the value IPLREQ if there is a match.
6. If no match has occurred yet, CDEMATCH checks for the abend code SA22 and returns the value OPERCANCEL if there is a match.
7. Further processing in the automation procedure depends on whether a matching code definition has been found and, if so, on the returned value of the matching code definition.

When calling CDEMATCH, the ENTRY parameter is not coded and defaults to JES2. This default occurs only if AOCQRY was previously called and task global variable SUBSTYPE is properly filled in. Refer to "AOCQRY" on page 40 for more information.

CHKTHRES

Purpose

The CHKTHRES command checks the number of errors recorded in the automation status file against a preset error threshold. It also supports recording of the error date and time in the automation status file.

CHKTHRES searches the automation control file for the applicable threshold for a specific resource. It then obtains the error status information from the automation status file and determines, based on the thresholds, whether any of the three definable thresholds are exceeded. If a threshold is exceeded, an error message is issued and an appropriate return code is generated.

Syntax

Parameters are positional.



Parameters

resource

The name of the resource that thresholds should be checked for. This value can be a subsystem name, a subsystem minor resource such as CICS.TRAN.APPL1, a generic MVS component name such as MVSESA.SMFDUMP, or any name. If the first qualifier of the specified resource name is not the name of a defined subsystem or MVSESA, the resource name is prefixed with MVSESA.

The resource name, including any prefix that might be added, is limited to a length of 64 characters.

This parameter is required.

NEW

If thresholds are defined for the specified resource name, an error timestamp is added to the error status information and then thresholds are checked.

CHECK

Thresholds are checked based on the existing error information.

CHKTHRES

COMMAND

This parameter determines the messages that CHKTHRES issues when the infrequent, frequent, and critical error thresholds are exceeded.

Specify this parameter if the automation procedure that uses CHKTHRES issues commands when an error threshold is exceeded.

NOMSG

This parameter determines that no messages are to be issued when the infrequent, frequent, and critical error thresholds are exceeded.

Restrictions and Limitations

None.

Return Codes

- 0 No threshold is exceeded
- 1 Infrequent threshold is reached
- 2 Frequent threshold is reached
- 3 Critical threshold is reached
- 4 Incorrect parameters were used in the call
- 5 Timeout or other error occurred

Messages

If CHKTHRES determines that a defined threshold level has been exceeded, it issues an appropriate message that depends on:

- The exceeded threshold level
- The type of resource that the threshold has been checked for
- The COMMAND parameter

If the COMMAND parameter was specified when calling CHKTHRES, message AOF589I, AOF588I, or AOF587I is issued.

Otherwise, if the COMMAND parameter is not specified and CHKTHRES is called to check the threshold for a subsystem, message AOF579I, AOF578I, or AOF577E is issued.

In all other cases (that is, when it is the threshold for anything other than a subsystem), CHKTHRES issues message AOF503I, AOF502I, or AOF501E.

All of these messages are captured with a severity that correlates to the exceeded threshold level, as follows:

Threshold exceeded	Severity
Infrequent	UNUSUAL
Frequent	IMPORTANT
Critical	CRITICAL

If necessary, you can change the assigned severity with a code definition for the message ID CAPMSGs as follows:

Code 1

Message ID

Code 2

Job name of subsystem

Code 3

*

Value Returned

Severity (This is required.)

Usage

- CHKTHRES accesses the automation policy to check the threshold definitions, and the automation status file to check the current error status information of the resource.
- CHKTHRES is used primarily to track error conditions that can be repetitive. By tracking the errors, operators can be notified of the repetitive error situation before it causes problems.

SA z/OS tracks a minimal number of situations, such as application abends, SPOOL shortages, and problems that cause full LOGREC conditions. It only tracks specific error messages when there are threshold definitions for the related minor resource, such as *subsystem.msgid* or *MVSESA.component*, and when commands are defined for those error messages.

- If no thresholds are defined for the resource itself, CHKTHRES searches in a predefined sequence to find the appropriate default threshold definitions. The search sequence depends on whether the resource name specifies a major resource, such as TSO, or a minor resource, such as TSO.IKT010D or MVSESA.SMFDUMP.

For major resources that are defined as subsystems, the default threshold definitions for applications or the system apply.

For the following minor resources, the default threshold definitions for MVS components or the system apply:

- MVSESA.MVSDUMP
- MVSESA.SMFDUMP
- MVSESA.LOGREC
- MVSESA.SYSLOG
- MVSESA.LOG

When searching for default threshold definitions for any other minor resources, the resource name is consecutively truncated up to the first two name qualifiers. For example, if no thresholds are defined for the minor resource CICS.TRAN.APPL1, defined thresholds for the truncated resource name CICS.TRAN are checked.

Example

This example shows the relationship between a CHKTHRES call in an automation procedure and thresholds defined in the automation policy database. The example involves thresholds set for the TSO subsystem. The automation procedure checks the thresholds by calling CHKTHRES.

The thresholds are defined in the automation policy database on the Thresholds Definition panel of the customization dialogs for the TSO subsystem.

The automation procedure to call CHKTHRES is:

```
/* REXX CLIST to check thresholds when a TSO error occurs          */
/* Check whether automation allowed and set TGLOBALs              */
'AOCQRY TSO AUTOMATION'
:
:
'CHKTHRES TSO NEW'
Select
```

CHKTHRES

```
    When rc = 0 Then Do
/*   perform actions required if no thresholds are exceeded   */
    :
    End
    When rc = 1 Then Do
/*   perform actions required if infrequent thresholds are exceeded */
    :
    End
    When rc = 2 Then Do
/*   perform actions required if frequent thresholds are exceeded   */
    :
    End
    When rc = 3 Then Do
/*   perform actions required if critical thresholds are exceeded. */
    :
    End
    Otherwise Do
/*   otherwise, an error occurred, RC=4/5, log error message   */
    :
    End
End
Exit
```

If, for example, the following thresholds settings are in effect for TSO:

- A critical threshold is defined as 8 errors occurring in 2 hours
- A frequent threshold is defined as 4 errors in 4 hours
- An infrequent threshold is defined as 4 errors in 8 hours

The example automation procedure performs the following processing steps:

1. If the automation procedure contains 'CHKTHRES TSO NEW' the time stamp when the error occurred is added to the automation status file, and the thresholds are checked.
2. Upon return to the automation procedure, the rc special variable is checked. If the value indicates that the critical threshold has been exceeded, the automation procedure should stop recovery to be consistent with the message that is issued by CHKTHRES.

FWDMMSG

Purpose

You can invoke the FWDMMSG command from the NetView automation table (AT) to forward messages from a remote system to a focal point system. By defining entries in the remote AT that invoke FWDMMSG you can:

- Trap messages that you are interested in
- Assign specific message classes to those messages
- Forward the messages to the focal point system

Messages are received by focal point notification operators who are defined to receive messages of the assigned classes.

Syntax



Notes:

- 1 Up to 10 classes may be specified. Classes should be separated by blanks.

Parameters

class

The message notification classes that are to be assigned to the message. You should specify at least one message class. If you do not specify a class, the message is sent to the authorized receiver of the GATOPER autotask. You can specify up to ten blank-delimited message classes. There are no default message classes.

SA z/OS notification classes are described in IBM Tivoli System Automation for z/OS Messages and Codes. You can define your own message classes using the AUTO MSG CLASSES policy item in the customization dialogs.

Note: The classes that you assign here must match those that you specify using the NOTIFY OPERATORS policy item in the customization dialogs.

SELF

This parameter sends the message to the appropriate notification operators on the issuing system if FWDMSG is invoked on a system that does not have a defined focal point. If FWDMSG is invoked on a system that does have a defined focal point, SELF is ignored.

MSG

The message text used for this message. If not coded, the messages in the message buffer are used. This parameter is valid for single-line messages only.

Restrictions and Limitations

- A triggering delete operator message will not be forwarded to the focal point.
- Do not use the MSG parameter for multiline messages.
- When FWDMSG is called from the NetView automation table, the message to be processed is in the message buffer. When FWDMSG is called from a command processor or other automation routine, the message text from the MSG parameter is treated as the entire message to be forwarded, including the message ID.
- If invoked with a pipe, all messages in the pipe are forwarded to the focal point as separate messages.

Return Codes

- 0 Automation procedure processed correctly.
- 1 Processing error was encountered.

When the MSG parameter is used for a multiline message, the following message is issued:

FWDMSG

AOF0131 SPECIFIED OPERAND MSG= INVALID FOR *msgid* MLWTO

Usage

You can call the FWDMSG command from the NetView automation table.

Examples

Example 1

The following example sends individual messages for each line in the multiline response:

```
IF MSGID='IST075I' & DOMAINID = %AOFDOM%  
THEN EXEC(CMD('FWDMSG A1')ROUTE(ONE *));
```

Example 2

The following example sends all RACF® messages to ensure notification of security violations:

```
IF MSGID='ICH' . & DOMAINID = %AOFDOM%  
THEN EXEC(CMD('FWDMSG A2')ROUTE(ONE *));
```

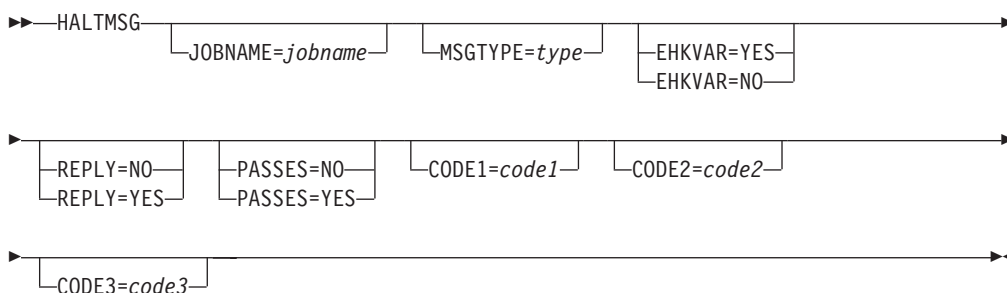
HALTMSG

Purpose

You can use the HALTMSG command to respond to a message by changing the status of an application to HALTED. HALTMSG calls the ISSUEACT command to also issue commands and replies that are defined in the automation policy for the ID of the processed message and for the HALTED status.

Typically, HALTMSG is called from the NetView automation table.

Syntax



Parameters

JOBNAME

The name of the job that the message is for. If not specified, the job name is taken from the message's job name field. You must supply a value for the job name if you are calling HALTMSG from a CLIST.

MSGTYPE

This parameter is used to search for command and reply entries to *subsystem/msgtype*-pairs in the automation control file, where *subsystem* is the subsystem name derived from the job name.

When a match occurs, the commands that are associated with the entries are issued. This is in addition to the command entries that are associated with the ENTRY-TYPE pair *subsystem*/HALTED.

If parameter MSGTYPE is not specified, the message identifier of the message that HALTMSG is called for is taken as the default.

EHKVAR

This parameter determines whether the tokens of the parsed message text are to be stored in task global variables EHKVAR0 through EHKVAR9 and EHKVART.

YES

The tokens of the triggering message are to be assigned to the task global variables EHKVAR*n*.

NO No values are to be assigned to the task global variables EHKVAR*n*.

REPLY

This parameter determines whether a defined reply is issued for a message that HALTMSG has been called for.

YES

A defined reply in the automation policy for the message that is being handled by HALTMSG is issued. REPLY=YES is assumed as the default if the message is a WTOR, otherwise the default is REPLY=NO.

NO A defined reply for a WTOR being handled by HALTMSG is not issued.

PASSES

Specifies whether passes are used to issue commands or replies (or both) that have been defined in the automation policy.

YES

PASSES=YES is passed to the ISSUEACT command.

NO PASSES=NO is passed to the ISSUEACT command.

CODE1=*code1*

CODE2=*code2*

CODE3=*code3*

These parameters are passed to the ISSUEACT command, where they are used to select defined commands and replies via code entries.

Restrictions and Limitations

- If HALTMSG is driven by a delete operator message, no action is taken in response to this message.
- HALTMSG will not affect an application that is being shut down.
- HALTMSG will not affect an application that is not in UP status.
- The application status is updated and the relevant commands are issued each time HALTMSG is run.
- Defined commands and replies are only issued in response to a message or a status change, if the recovery flag of the related minor resources of the application allows automation.
- If this command is called on a task other than the AOFWRK*xx* auto operator that is responsible for the subsystem, HALTMSG will schedule itself to that AOFWRK*xx* auto operator. The HALTMSG command will run asynchronously to the calling procedure. This means that when the calling procedure regains control, the status of the subsystem may not yet have changed.

HALTMSG

- Only messages for applications with known address space IDs are processed by HALTMSG.
The address space ID is not checked if HALTMSG is called from an automation procedure (CLIST), or if HALTMSG has been triggered by message BPXF024I.

Usage

You should normally call the HALTMSG command from the NetView automation table.

Applications can be put into HALTED status when something occurs that leaves them running with reduced function. Use HALTMSG to put an application into HALTED status, and ACTIVMSG (or the SETSTATE command dialog) to change the status.

If HALTMSG is called for a WTOR and it is not replied to, OUTREP is called to process the WTOR.

HALTMSG should run on the working operator of the subsystem that issued the message. Otherwise, the HALTMSG command will run asynchronously to the calling procedure. This means that when the calling procedure regains control, the status of the affected subsystem may not yet have changed.

All commands and replies that are triggered through HALTMSG have access to the SAFE called AOFMSAFE, which stores the message that caused the HALTMSG call.

Task Global Variables

EHKVAR0 through EHKVAR9 and EHKVART

When defining the commands in the automation control file to be issued by command HALTMSG, the variables &EHKVAR0 through &EHKVAR9 and &EHKVART can be used to be substituted by the tokens of the parsed message that has driven HALTMSG. &EHKVAR0 will be substituted by the message ID, &EHKVAR1 by the first token of the message text after the message ID, &EHKVAR2 with the second token and so forth. &EHKVART will be substituted by the trailing message text after the 9th token.

Examples

The following example shows how HALTMSG is called from the NetView automation table:

```
* IKT008I TCAS NOT ACCEPTING LOGONS
IF MSGID = 'IKT008I' & DOMAINID = %AOFDOM% THEN
EXEC( CMD('HALTMSG')
ROUTE(ONE %AOFOPGSSOPER%));
```

INGALERT

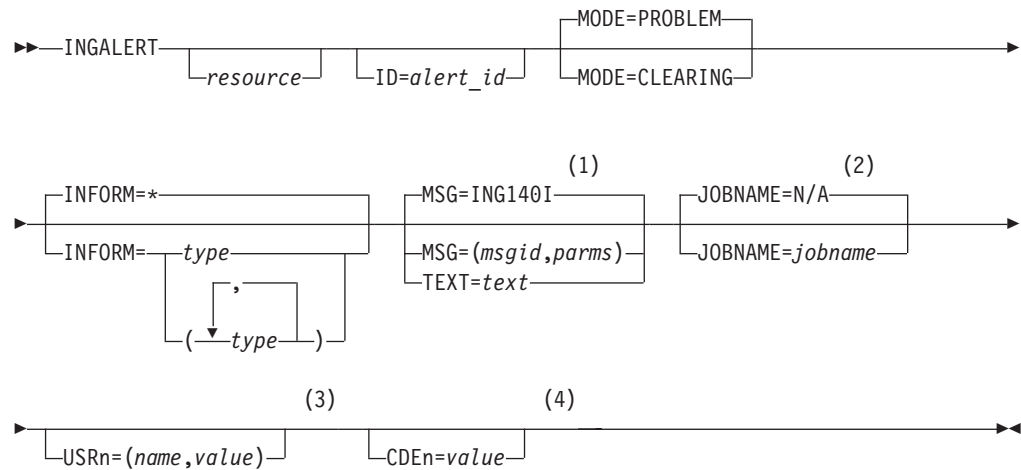
Purpose

The INGALERT command allows you to send alerts to event notification targets such as:

- System Automation for Integrated Operations Management (SA IOM)
- Tivoli Enterprise Console (TEC)
- Tivoli Netcool/OMNibus

- Tivoli Service Request Manager®
- A user-defined alert handler.

Syntax



Notes:

- 1 This is the default if INGALERT is not called from the NetView automation table.
- 2 This is the default if INGALERT is not called from the NetView automation table.
- 3 You can define USR1 - USR9.
- 4 You can define CDE1 - CDE9.

Parameters

resource

This can be:

- A fully-qualified resource name, in the form *name/type/sys* (where *type* is APL, APG or MTR)
- The text MVSESA (that is, the content of the AOFSYSTEM variable)
- A job or subsystem name

The subsystem name takes precedence over the job name.

If there is more than one resource with the given subsystem or job name, the corresponding local system resource is used. If there is no such resource on the local system, the first resource found is used.

If the resource name is not specified as a parameter, the value of the JOBNAME parameter is used to determine the resource name. If the JOBNAME parameter is not specified either, and INGALERT is called from the automation table, the job name is taken from the message's job name field. If no job name can be determined, MVSESA is used as the resource name.

- ID** This is the alert identifier for the resource. Refer to the "Enable Alerting" section in *IBM Tivoli System Automation for z/OS Customizing and Programming* for further details on the pre-defined alerts for SA z/OS. If

INGALERT

| INGALERT is called from the NetView automation table (AT), *alert_id* can
| be omitted, in which case the message ID of the triggering message is
| used.

| When you create clearing events by specifying `MODE=CLEARING`, you
| can set the `ID=*` to clear all outstanding events for the specified resource.

MODE

This specifies the type of created event.

PROBLEM

Creates a problem event. This is the default value.

CLEARING

Creates a clearing event only if the event is addressed to the EIF or
USR target. For other targets, no event is created.

INFORM

This specifies the notification target for the created event to be sent to. An
event is only sent to a specified target, if the event notification for this
target is enabled and if the target is included in the inform policy list of
the resource.

The value for the INFORM parameter can be specified as:

- The name of a supported event notification target. Supported targets are
IOM, EIF, TTT or USR
- A list of supported event notification targets, enclosed in parentheses.
- An asterisk, *, which means the list of all supported event notification
targets.

MSG This is the ID of a message defined in the NetView message catalog.

Note that if you enter an invalid message ID, the message AOF000I is sent
to the specified event notification targets.

The message can have variables &1 through &9 that are filled with
parameters that are specified after *msgid*. If a message parameter is omitted
the following defaults are used:

- &1** INGALERT
- &2** The date and time stamp when the message was generated
- &3** The alert ID that was specified or the default that was used
- &4** The resource that was specified or the default that was used
- &5** The system that INGALERT was called on

MSG cannot be used together with TEXT.

TEXT This is a text string that is to be used for the alert.

TEXT cannot be used together with MSG.

JOBNAME

This specifies the name of the job that caused the alert. If not specified, the
job name is taken from the message's job name field if INGALERT is called
from the automation table. Otherwise N/A is used as the default value.

| **USRn** Specifies the name and the value of a user field. n can be 1-9. User fields
| are passed on to notification targets EIF, TTT, and USR.

| **CDEn** Specifies the value to replace '&n' placeholders in the CDEMATCH data. n
| can be 1-9.

If neither MSG nor TEXT is specified and INGALERT is called from the AT, the text of the triggering message is used. Otherwise MSG=ING140I is used.

Note that the text or message text may be truncated by SA IOM.

Return Codes

- 0 All alerts were sent successfully. In addition, there might be alerts that have been ignored.
- 1 All alerts were ignored.
- 3 Besides alerts that were successfully sent, at least one alert could not be sent. In addition, there might be alerts that have been ignored.
- 4 Parameter error.
- 6 Environment check failed.
- 8 No alert could be sent. In addition, there might be alerts that have been ignored.

Restrictions and Limitations

SA z/OS must be fully initialized.

Usage

Alert transmission can be controlled as follows:

- Alerting can be enabled and disabled system-wide with the INGCNTL command.
- If an event notification target is specified in the Inform List field of the resource's policy, alerting is enabled; otherwise it is disabled.
- Alerting is disabled for a resource if code processing for it is unsuccessful or returns a value of IGNORE.

An event is only sent to the specified target if event notification for this target is not prevented by one of these control mechanisms.

SA z/OS does not keep track of alerts that have been sent. Once the data has been successfully delivered to the event notification target, responsibility for delivering the event is passed on to this target as follows:

- For IOM, the event is passed on to SA IOM via the peer-to-peer connection
- For EIF, the event is passed on to the message adapter service of the NetView event/automation service
- For TTT, the event is passed on to the Tivoli Directory Integrator server
- For USR, the event is passed on to the user defined alert handler

If an alert cannot be sent to the target a message is written to the netlog. No further attempts are made to deliver the alert.

Examples

Example 1:

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'));
```

INGALERT

This uses alert ID ABC123I and the complete message text of ABC123I as the alert text and sends the alert to all the targets on the inform list definition in the automation policy.

Example 2:

The following can be used from the command line or a CLIST:

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

This uses alert ID MYALERT and the specified alert text and send an alert to IOM only.

INGCLEAN

Purpose

The INGCLEAN command can be used to clean up the in-storage data model of the local system. The data model is compared to the currently loaded configuration files. Entries within the data model that have no corresponding entries in the configuration files are deleted. For entries that have a hard-coded default provided by System Automation, this default value is restored.

Syntax

```
▶▶—INGCLEAN—————▶▶
```

Parameters

There are no parameters. Any specified parameters will be ignored without notification.

Return Codes

- 0 Cleanup was successful.
- 4 Parameter Error.
- 6 Environment check failed.
- 8 An error occurred.

Restrictions and Limitations

In order to run INGCLEAN the configuration must be loaded and the configuration files must be accessible from disk.

INGCLEAN does not affect data for subsystems, subsystem defaults, application groups and monitor resources because they are cleaned up by the main configuration loader process.

INGCLEAN does not affect data for automation operators because work could be scheduled there and a cleanup would cause an immediate log off from the corresponding task.

For all other elements, INGCLEAN will remove all data if the element is not present in the currently loaded configuration files. This means especially that all

data that has been added via ACF REQ=REPL is deleted. Note that INGCLEAN does not restore the values from the configuration files for data that has been modified via ACF REQ=REPL.

INGCLEAN always runs on the local system where it is invoked.

Usage

You can run INGCLEAN from the policy, the automation table, an automation script or directly from the command line. It is recommended to run INGCLEAN as an environment setup exit if you want it to run everytime you load or refresh your configuration from the command line for an instantaneous regular cleanup timer.

Examples

If you want to run INGCLEAN on every configuration refresh, specify it as the first environment setup exit.

```

COMMANDS  HELP
-----
System Information                                Top of data
Command ==> _____
Entry Type : System                               PolicyDB Name : USER_PBD
Entry Name : SYS3                                 Enterprise Name : USER_ENTERPRISE
                                                More:      +
Operating system . . . : MVS
Image/System name. . . : SYS3

The following specifications are for MVS systems only:
Primary JES. . . . . JES2           Primary JES2/JES3 subsystem name
System Monitor Time. . . 00:59       Time between monitor cycles (hh:mm or NONE)
Gateway Monitor Time . . 00:15       Time between monitor cycles (hh:mm or NONE)
Automation Table(s) . . INMSG01

-----
SDF Root Name. . . . . NetView automation table members
Exit name(s) . . . . . Root of system's SDF tree
                        INGCLEAN MYEXIT
                        Environment setup user exit names
USS automation path. . .
/usr/lpp/ing/ussauto/lib

-----
SDF Root Name. . . . . NetView automation table members
Exit name(s) . . . . . Root of system's SDF tree
                        Environment setup user exit names
USS automation path. . .
/usr/lpp/ing/ussauto/lib

-----
SA NetView Domain. . . . System Automation UNIX installation
Network NetView Domain . NetView domain ID of SA z/OS
                        NetView domain ID of network automation

NMC Focal Point Communication specification:
Heartbeat Interval . . . . 5           (1 - 60 minutes)
Missing Heartbeat Delay. . 30         (1 - 3600 seconds)
F1=HELP      F2=SPLIT   F3=END      F4=RETURN   F5=RFIND    F6=RCHANGE
F7=UP        F8=DOWN    F9=SWAP    F10=LEFT   F11=RIGHT   F12=RETRIEVE

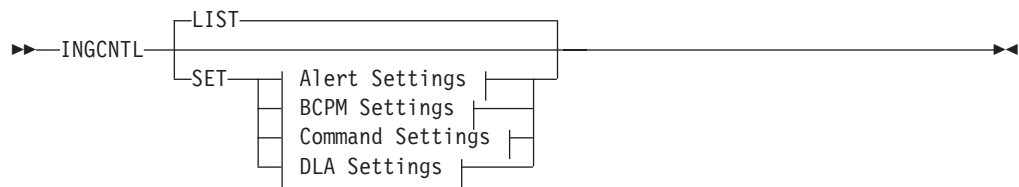
```

INGCNTL

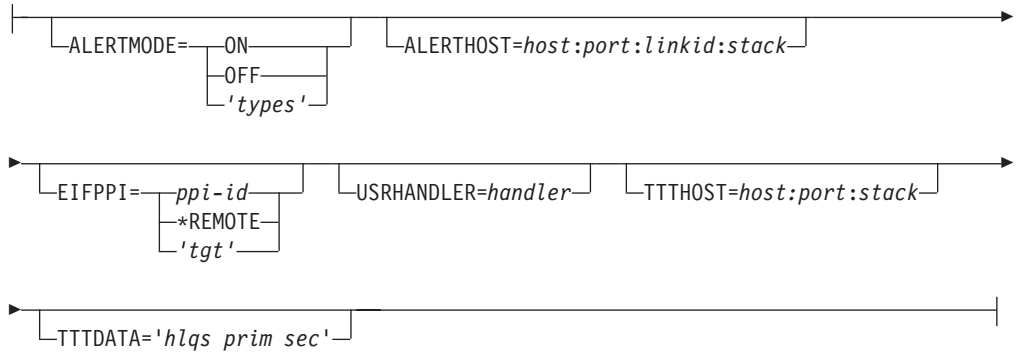
Purpose

The INGCNTL command is used to control various SA z/OS settings.

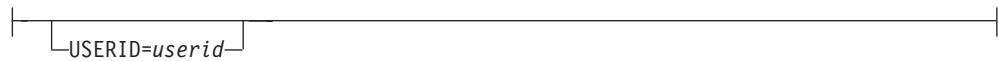
Syntax



Alert Settings:



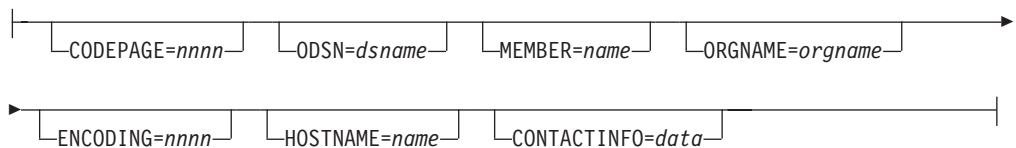
BCPM Settings:



Command Settings:



DLA Settings:



Parameters

LIST This lists all of the supported settings. This is the default.

SET This allows you overwrite one or more settings.

You can use '' to delete the value of any of the INGCNTL settings, for example, INGCNTL SET ORGNAME=''

ALERTMODE

This sets the alerting mode:

ON Enables alerting for all event notification targets.

OFF Disables alerting for all event notification targets.

'types' Enables alerting for the specified event notification targets. The value of *types* must be a blank-separated list of the supported event notification targets IOM, EIF, TTT, or USR.

ALERTHOST

This sets the properties of the connection to the SA IOM server.

host The host name or IP address of the SA IOM server to be used for the notification.

Note: IPv6 addresses must be enclosed in square brackets ([. . .]).

port The port of the SA IOM server to be used for the notification.

linkid The link ID to be used for the connection to the SA IOM server. It can be up to 8 characters long.

stack IP stack name to be used. If not specified, the default of the SOCKET command is used.

Note that omitting the colon separators restores the original settings for the connection.

EIFPPI

Specifies the PPI receiver name of the NetView Event/Automation Service (E/AS) message adapter service, which forwards alerts to an EIF target such as the Tivoli Enterprise Console or Tivoli Netcool/OMNIBus. Alternatively, "*REMOTE" can be specified to forward the EIF alert to a remote system (outside the local sysplex) for passing the event to the E/AS service.

tgt is the name of the system or its domain id used as a hub. If omitted, the current focal point is used.

Note: If *tgt* is specified with *REMOTE you must use single quotes, for example INGCNTL...EIFPPI='*REMOTE SYS1....'

USRHANDLER

Specifies the name of the automation procedure to be executed to handle events destined for the target USR. See the AOFEXALT member in ING.SINGSAMP for an example.

TTTHOST

Specifies the properties for the connection to the Tivoli Directory Integrator (TDI) server.

host The host name or IP address of the TDI server that is to be used as the destination for trouble tickets.

port The port of the TDI server that is to be used as the destination for trouble tickets.

stack IP stack name to be used. If not specified, the default of the SOCKET command is used.

Note: IPv6 addresses must be enclosed in square brackets ([. . .]).

TTTDATA

Specifies the data set characteristics for the trouble ticket detail data, where:

hlqs The prefix for the data set name, consisting of one or more qualifiers, with a maximum length of 16 characters. This value must follow the naming conventions for a valid data set name. The name of the data set will be *hlqs.domainid.Ddate.Ttime.Ccounter*.

prim The number of cylinders for the primary allocation.

sec The number of cylinders for the secondary allocation.

USERID

Specifies the MAXIMO user ID for BPCM Web Service authentication.

COMMAND_LOGGING

This sets command logging via message AOF705I for several SA z/OS commands. Message AOF705I lists all the parameters that have been specified together with the user ID of the person or autotask that issued the command.

YES Enables command logging.

NO Disables command logging.

CODEPAGE

Specifies the encoding codepage — it is the one that NetView uses. The default is 1047.

You must specify this parameter if you are running with a different codepage. Failure to do so will result in the generation and downloading of a corrupt Identity Markup Language (IdML) book that Tivoli Application Discovery Dependency Manager (TADDM) cannot load.

ODSN

Specifies the name of the output data set. The data set must be a pre-allocated (catalogued) PDS with attribute VB=3000. The name must be fully qualified, with or without surrounding quotation marks. The user ID that NetView is running under must have UPDATE access to it.

MEMBER

Specifies the name of the member that will contain the IdML data. The default is INGBOOK.

Reserved member name is @CHCKSUM.

ORGNAME

Specifies the name of the organization. The default is to take the default name from the IBM Tivoli Change and Configuration Management Database (CCMDB).

ENCODING

Specifies the encoding option. Valid values are EBCDIC, ASCII and UTF-8. The default is UTF-8.

HOSTNAME

Specifies the name of the host of the management software system (that is, SA z/OS). It is used to address SA z/OS. If specified, it takes precedence over a discovered host name.

CONTACTINFO

Provides details of ports and security keys that are needed to establish a session with NetView over TCP/IP when used in conjunction with the host name.

Return Codes

- 0 Normal completion. Settings have been applied.
- 4 Invalid parameters were specified.
- 8 Command failed.

Usage

INGCNTL behaves like an operator command. The output from INGCNTL LIST command takes the form of a correlated multiline message ING149I.

The first line of the multiline message is a header text. Output of the settings begins on line two. Each setting is in a separate line and in the format keyword: value.

Use INGCNTL in a PIPE to capture and analyze (or suppress) the generated message or as an operator NCCF command.

Examples

To list the current settings specify:

```
INGCNTL LIST
```

The output produced is similar to the following:

```
ING149I LIST CONTROL SETTINGS
      ALERTMODE: OFF
      CODEPAGE: 1047
      ORGNAME: My Organization
      ODSN: USER.DLA
```

Note that unset values that have no defaults are not listed.

To enable alerting for IOM and EIF, specify:

```
INGCNTL SET ALERTMODE='IOM EIF'
```

To set the connection properties for IOM specify, for example:

```
INGCNTL SET ALERTHOST=MYIOMSRVR:4711:SA2IOM
```

To set the connection properties for IOM using an *IPv4* address, specify, for example:

```
INGCNTL SET ALERTHOST=10.0.0.3:4711:SA2IOM
```

To set the connection properties for IOM using an *IPv6* address, specify, for example:

```
INGCNTL SET ALERTHOST=[::0AFF:7F01]:4711:SA2IOM
```

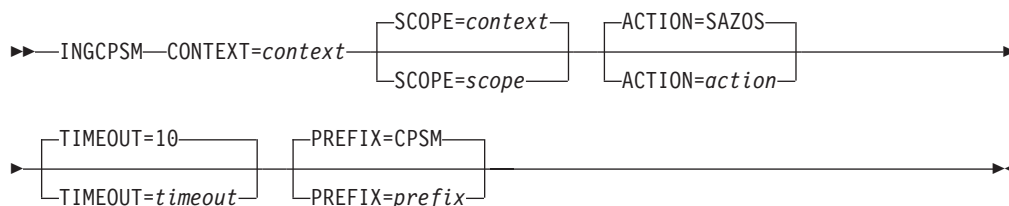
Note: You can invalidate the ALERTHOST settings for an instance by removing the colons. This makes it easy to restore to the original settings.

INGCPSM

Purpose

The INGCPSM command returns status information for a CICSplex[®] System Manager (CICSplex SM) object. This data is returned in ING150I messages. INGCPSM should be run in a virtual operator station task (VOST).

Syntax



Parameters

CONTEXT

This identifies the context for the command. *context* must be the name of a CICSplex and can be 1–8 characters long. This parameter is required.

Note that a CICSplex SM address space (CMAS) name is not allowed.

SCOPE

This qualifies the CONTEXT option. *scope* can be:

- The 1- to 8-character name of the CICSplex itself
- A CICS system or CICS system group within the CICSplex
- A logical scope, as defined in a CICSplex SM resource description (RESDESC)

If *scope* is not specified the same value that is defined for the context is used.

ACTION

This limits the events that INGCPSM handles to those that are defined with a certain ACTION DEFINITION name.

If *action* is not specified SAZOS is used as the default.

TIMEOUT

This is the number of seconds to wait and collect events before ING150I messages are produced. This is also the interval during which monitor resources that require initial monitoring are determined. The valid range is 1–30 seconds.

If *timeout* is not specified 10 seconds is used as the default.

PREFIX

This is the 1-4 character string that prefixes a monitored CPSM object in the monitor resource. This is used to identify CPSM-related MTRs for initial monitoring.

If *prefix* is not specified CPSM is used as the default.

Return Codes

None.

Restrictions and Limitations

SA z/OS must be fully initialized.

| INGCPSM requires CICSplex SM to be installed and ready to use. Appropriate
| definitions must have been made within CPSM. See the step “Installing CICSplex

SM REXX API” in the chapter “Installing SA z/OS on Host Systems” in *IBM Tivoli System Automation for z/OS Planning and Installation*.

Usage

Use the INGVSTRT command to run INGCPSM in a VOST, see “INGVSTRT” on page 158.

Examples

To start INGCPSM and monitor CICSplex CICPLX1, enter the following as a start command in the VOST management APL:

```
INGVSTRT SYNC,INGCPSM CONTEXT=CICPLX1
```

To override the defaults for ACTION and TIMEOUT specify:

```
INGVSTRT SYNC,INGCPSM CONTEXT=CICPLX1,ACTION=MYACT,TIMEOUT=30
```

INGDATA

Purpose

The INGDATA command returns detailed information that the automation manager maintains for the specified resources. The data is returned as a multiline message, one line for each resource.

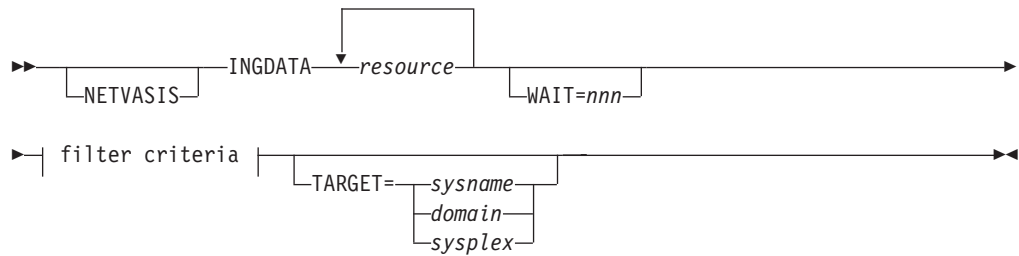
The format is as follows:

Byte	Length	Description
1	11	Name of resource
14	3	Resource type, for example, APG or APL
19	8	Name of system hosting resource
29	11	Observed status
41	12	Desired status
54	10	Automation status
65	4	Automation flag
70	4	Hold flag
75	48	Description
125	10	Start type
137	8	Stop type
147	8	Service period name
157	8	Trigger name
167	12	Compound status
181	10	Startability status
193	8	Resource nature (group type)
203	8	Category
213	10	Subcategory
224	12	Health status
237	8	Jobname
246	40	Inform list

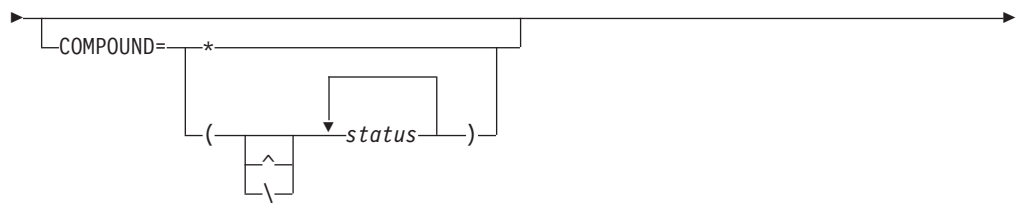
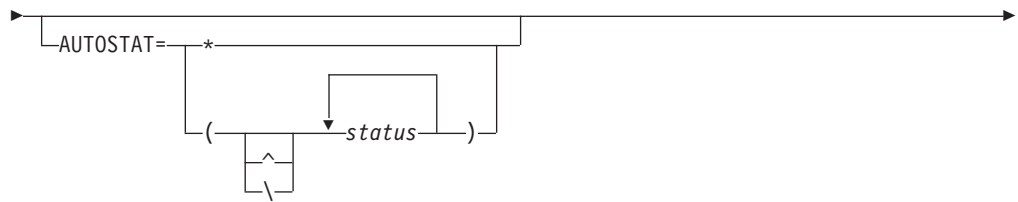
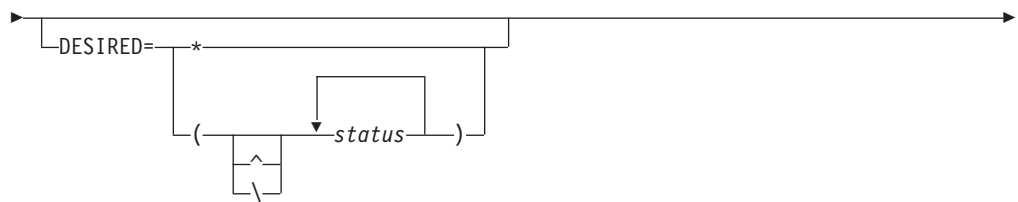
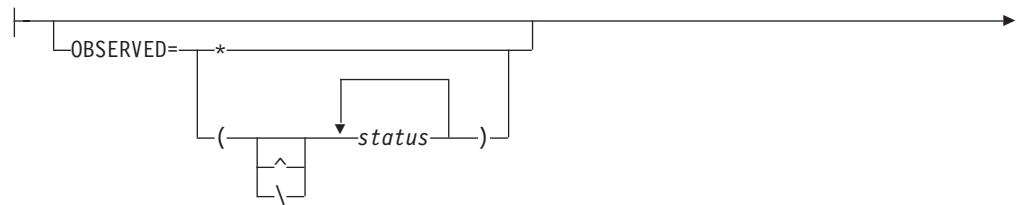
INGDATA

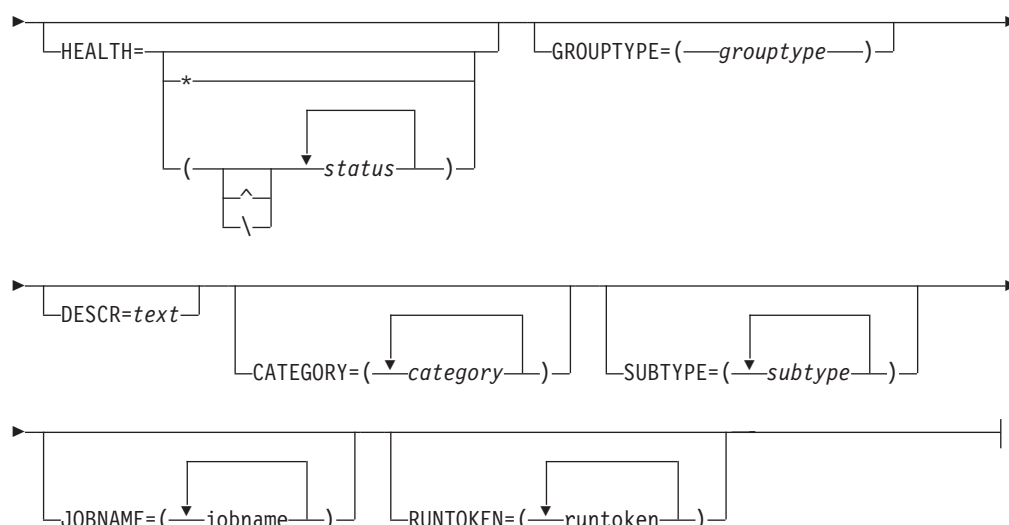
Byte	Length	Description
287	3	Runmode qualification
291	8	Desired default status

Syntax



filter criteria:





Parameters

NETVASIS

Prefix the INGDATA command with NETVASIS if you want to pass the description text in lower or mixed case.

resource

Specifies the name of the resource (or resources) to be displayed. The format is name/type[</system>]. It can be a list of names.

The resource names must be separated by a blank. Asterisks (*) can be used as wildcard characters.

WAIT

Specifies the number of seconds to wait before reporting that a timeout occurred if the automation manager does not provide the requested data. The maximum time interval is 999 seconds.

If omitted, the time interval is 30 seconds.

filter criteria

The filter criteria to be applied prior to displaying the data. The following values can occur:

OBSERVED

Specifies the observed statuses to be displayed. The statuses must be separated by a blank. It can be abbreviated, for example, to AV for available. If '^' or '\' is used, all statuses except the ones you specify are displayed.

DESIRED

Specifies the desired statuses to be displayed. The statuses must be separated by a blank. It can be abbreviated, for example, to AV for available. If '^' or '\' is used, all statuses except the ones you specify are displayed.

AUTOSTAT

Specifies the automation status to be displayed. The statuses must be separated by a blank. It can be abbreviated, for example, to ID for idle. If '^' or '\' is used, all statuses except the ones you specify are displayed.

INGDATA

COMPOUND

Specifies the compound status. The statuses must be separated by a blank. It can be abbreviated, for example, to SA for satisfactory. If '^' or '\' is used, all statuses except the ones you specify are displayed.

CATEGORY

Specifies the category that the resource belongs to. More than one value can be specified.

SUBTYPE

Specifies the subtype of the resource. More than one value can be specified.

DESCR

Specifies the text string used as a filter. The text can contain wildcards. An asterisk (*) matches a string of arbitrary length and a percentage sign (%) matches a single character. The DESCR parameter is case-sensitive. The text string must be enclosed in single or double quotation marks or parentheses() to maintain the case-sensitivity of the entry.

GROUPTYPE

Specifies the type (nature) of the resource group. More than one value can be specified.

HEALTH

Specifies the desired health statuses to be displayed. The statuses must be separated by a blank. It can be abbreviated, for example, to NO for normal. You can also specify an asterisk (*) to reset the current filter setting, for example, INGFLT HEALTH=*. If '^' or '\' is used, all statuses except the ones you specify are displayed.

JOBNAME

The jobname assigned to the resource. More than one jobname can be specified. Wildcards are supported.

RUNTOKEN

The runtoken assigned to the resource. More than one runtoken can be specified. Wildcards are supported.

TARGET

For information on the TARGET parameter, refer to *IBM Tivoli System Automation for z/OS Operator's Commands*.

Return Codes

- 0 Okay.
- 1 An error occurred.
- 2 SA z/OS has not fully initialized.

Restrictions and Limitations

SA z/OS must be fully initialized.

Usage

The INGDATA command should be used in a NetView PIPE statement.

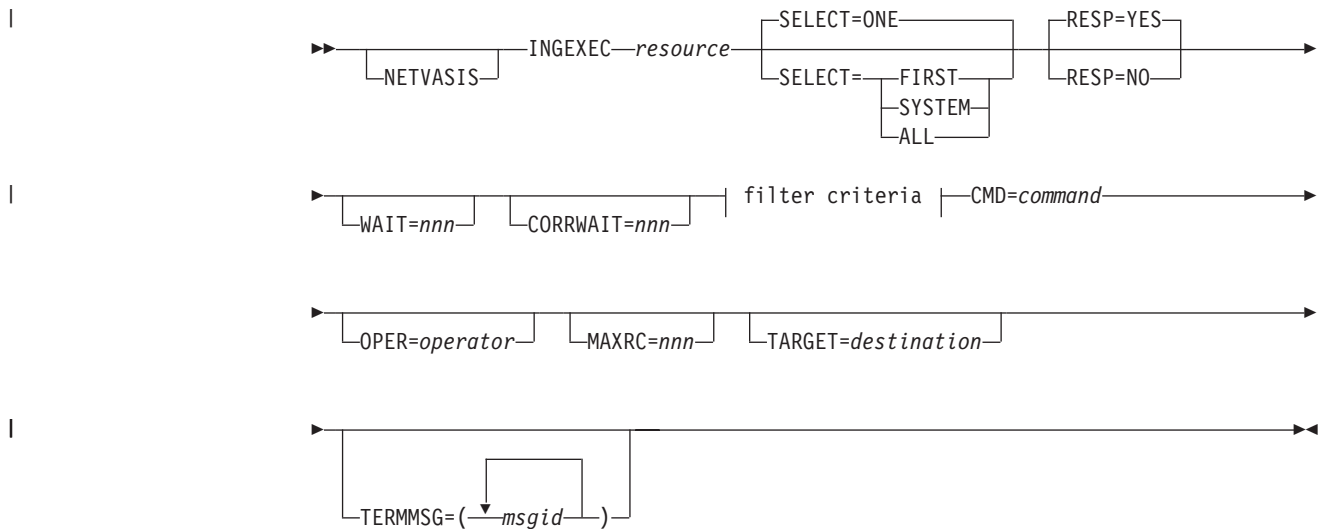
INGEXEC

Purpose

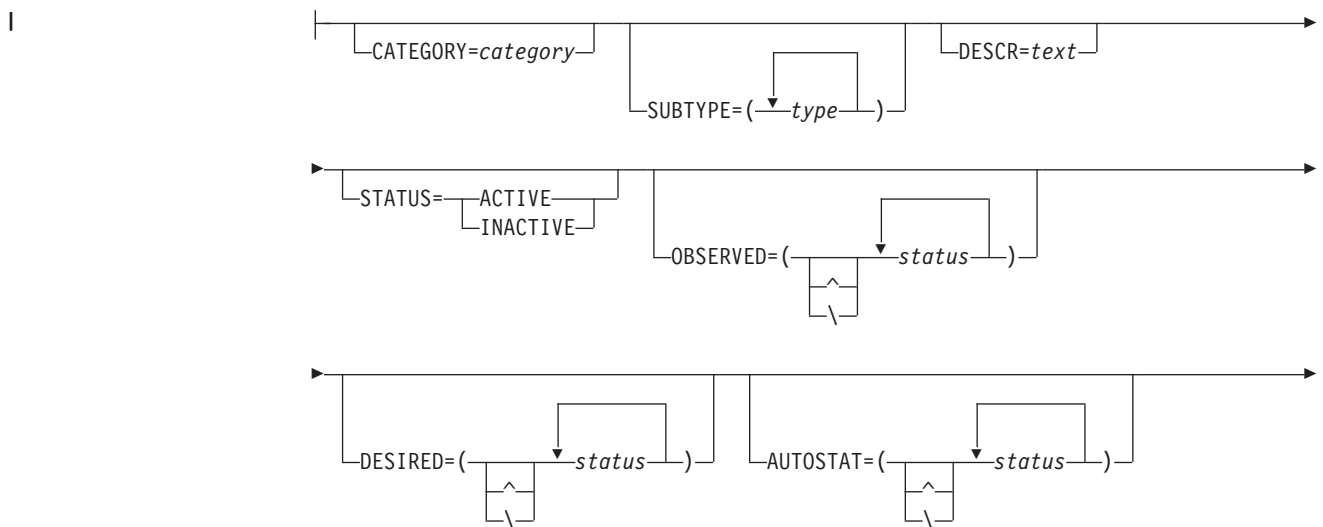
The INGEXEC command can be used to process the specified command on the system where the specified resource resides. The output of the command execution is collected and can be automatically returned to the caller.

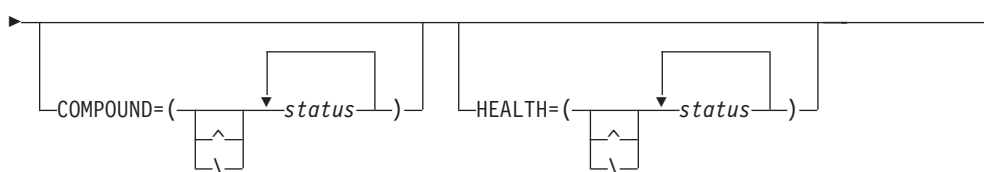
The INGEXEC command operates sysplex-wide. The INGEXEC command interrogates the automation manager to determine the list of resources affected.

Syntax



filter criteria:





Parameters

NETVASIS

Prefix the INGEXEC command with NETVASIS if you want to pass the command and/or the description text in lower or mixed case.

resource

The resource name that is used to determine the system that the command should be routed to. It can be one of the following:

- The resource name in automation manager format. If an application group is specified its group members are taken. Note that a resource name in automation manager notation has the format `resname/type[/system]`. The resource name can contain wildcards, such as `TSO*/APL/*` or `CICS%%G/APG`.
- The subsystem name. If the name that is specified for a resource does not contain a slash (/) it is considered to be a subsystem name. The subsystem name can contain wildcards, such as `TSO*` or `*DB2`.

Note: When the INGEXEC command is specified in the policy database, the subsystem symbols are resolved at the time the INGEXEC command is executed and *not* when the command specified in INGEXEC is processed. To ensure that the substitution is done when the command specified in INGEXEC is executed, prefix the variable name with a number sign character (#) instead of the standard ampersand character (&), for example, `#SUBSJOB`.

SELECT

Specifies which resource is used when determining where to send the command:

- ALL** All resources that match the filter criteria.
- FIRST** The first resource in the list of eligible resources. The resource list is sorted in alphabetical order.
- ONE** Only one resource is allowed to match the filter criteria. This is the default. If more than one resource matches the criteria, the command is rejected.
- SYSTEM** One resource per system is allowed to match the filter criteria.

RESP

Specifies how to handle the command output:

- YES** The output of the command is returned to the caller.
- NO** The output of the command is queued with the NetView CMD LOW command on the target systems.

WAIT

Specifies the number of seconds to wait before reporting that a timeout occurred if the automation manager does not provide the requested data. The maximum time interval is 999 seconds. If omitted, the time interval is 30 seconds.

CORRWAIT

Specifies the CORRWAIT value (in seconds) to be used when INGEXEC uses the NetView PIPE command to submit the command to be executed. The CORRWAIT PIPE stage is necessary to trap asynchronous command output. The maximum value is 999 seconds.

filter criteria

The following filter criteria can be optionally specified:

CATEGORY

The category of the resource.

SUBTYPE

The subcategory of the resource. More than one subcategory can be specified. Wildcards are supported.

DESCR

Specifies the text string as a filter. The text can contain wildcards. An asterisk (*) matches a string of arbitrary length and a percentage sign (%) matches a single character. The DESCR parameter is case-sensitive. The text string must be enclosed in single or double quotation marks or parenthesis to maintain the case-sensitivity of the entry.

STATUS

The status that the resource must be in to be considered. It can be one of the following:

ACTIVE The observed status of the resource must be AVAILABLE.
INACTIVE The observed status of the resource must be either SOFTDOWN, HARDDOWN, SYSGONE, or UNKNOWN.

OBSERVED

Specifies the observed statuses that the resource must be in to be eligible. The statuses must be separated by a blank. It can be abbreviated, for example, to AV for available. If '^', or '\' is used, all statuses except the ones you specify make the resource eligible.

DESIRED

Specifies the desired statuses that the resource must be in to be eligible. The statuses must be separated by a blank. It can be abbreviated, for example, to AV for available. If '^', or '\' is used, all statuses except the ones you specify make the resource eligible.

COMPOUND

Specifies the compound statuses that the resource must be in to be eligible. The statuses must be separated by a blank. It can be abbreviated, for example, to SA for satisfactory. If '^', or '\' is used, all statuses except the ones you specify make the resource eligible.

AUTOSTAT

Specifies the automation statuses that the resource must be in to be eligible. The statuses must be separated by a blank. It can be abbreviated, for example, to ID for idle. If '^', or '\' is used, all statuses except the ones you specify make the resource eligible.

HEALTH

Specifies the health statuses that the resource must be in to be eligible. The statuses must be separated by a blank. It can be abbreviated, for example, to MI for minor. If '^', or '\' is used, all statuses except the ones you specify make the resource eligible.

INGEXEC

CMD

The command to be executed. It can contain &SUBxxxx variables that are resolved using the appropriate settings of the subsystem on the target system.

OPER

The automated operator on the target system where the command is to be processed.

MAXRC

The maximum return code accepted for command processing. If the command return code is higher processing is aborted. For multiple resources, the processes run sequentially and a process is terminated if a return code is greater than the value specified here.

TERMMSG

Specifies the message ids that terminate the message collection. More than 1 message id can be specified separated by a blank character. If more than 1 message id is specified, they must be enclosed in parenthesis or quotes.

TARGET

The systems where the INGEXEC command should run. The INGEXEC command is a sysplex-wide command. Therefore you do not need to specify the TARGET parameter when you want processing within the local sysplex. Specify only one system for each remote sysplex. Specify *FP when the command should be routed to the focal point.

Return Codes

- 0 Command processed successfully.
- 1 An error occurred.
- 2 Parsing Error.

Restrictions and Limitations

SA z/OS must be fully initialized.

Any color attribute that is associated with the output messages of the command that is being executed are ignored. This also applies to any screen manipulation actions.

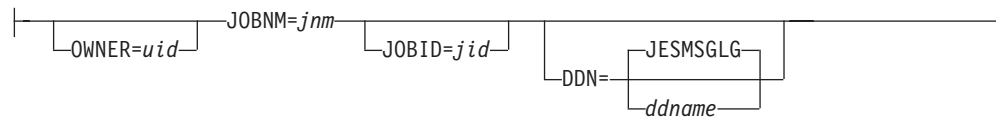
Examples

Consider a setup with three basic DB2 application groups that each have three DB2 applications: DB2MSTR, DB2IRLM, and DB2DIST. All three applications in DB2GRP/APG/SYS1 are active, DB2DIST in DB2GRP/APG/SYS2, and all three applications in DB2GRP/APG/SYS3 are inactive.

Now consider the result of the following:

1. `INGEXEC DB2GRP/APG/* STATUS=ACTIVE SUBTYPE=MSTR SELECT=ONE CMD='...'`
In this example, both DB2MSTR/APL/SYS1 and DB2MSTR/APL/SYS2 are selected. However, because SELECT=ONE is specified the command is denied.
2. `INGEXEC DB2/APG STATUS=ACTIVE SUBTYPE=DIST SELECT=ALL RESP=NO CMD='...'`
In this example, DB2DIST/APL/SYS1 is selected. It is the only resource with subtype=DIST that is active. The command is routed to SYS1. Subsystem symbol (&SUBxxxx) substitution takes place using the subsystem settings of DB2DIST on SYS1. Because RESP=NO is specified, SA z/OS does not wait for command completion.

dsname:



Parameters

DDN

Defines the ddname of a valid JES spool output data set:

JESMSG LG JES message log. This is the default.

ddname The ddname of any other spooled output data set.

INTERVAL

Defines the monitoring interval for the indicated ddname in seconds. The minimum interval is 1 second, the maximum is 3600 seconds.

JOBID

Defines the job identifier, for example, JOB08450. The value can be up to 8 characters long.

Note: The command is rejected when the parameter is omitted and multiple job ids are found for the particular job name.

JOBNM

Defines the name of the job. The value can be up to 8 characters long.

OWNER

Defines the owner of the job. The value can be up to 8 characters long.

Note: The command is rejected when the parameter is omitted and multiple jobs are found for the particular job name with different owner ids.

START

Start the joblog monitoring of the indicated log.

STATUS

Show the status of the joblog monitoring task and its monitored resources.

STOP

Stop the joblog monitoring of the indicated log.

SUSPEND

Suspend the joblog monitoring. All opened data sets are closed and deallocated. Finally, the task is terminated. However, all information to continue the preceding monitoring is kept. When the task is restarted it reopens each data set, positions to the last record read, and processes each new accumulated message.

Note: When you are no longer interested in joblog monitoring at all use the NetView command STOP to terminate the task. All data sets are closed and all storage is released.

Restrictions and Limitations

The command is available only when JES2 is the primary subsystem.

When the monitoring task is restarted after it was suspended the joblog monitoring continues with the following limitations:

- The output of jobs that have finished before the monitoring task has been restarted is lost unless the output is still held on the output queue.
- A job is not monitored when it has been started between the time frame of the suspension of the task and its restart.

For manually monitored jobs you cannot specify any filter criteria to limit the data that is passed to automation. This means any line of data except an empty line which is generally excluded is forwarded to the message automation. The automation is always performed by the autotask LOGOPER.

Usage

The start and stop commands should be used only for jobs which are not controlled by SA z/OS.

The suspend command is used to suspend the joblog monitoring. The accumulated output of all monitored jobs is not processed before the task has been restarted. To restart the task use the NetView command START TASK=INGTJLM.

The query command is used to determine the current status of all monitored jobs. This command can be issued in a NetView PIPE.

Examples

The following sample shows you how to monitor a spooled data set other than the default data set of a job which is not controlled by SA z/OS and which was started prior to the command

```
INGJLM START JOBNM=HIRZJLMS INTERVAL=10 DDN=AAAZOUT
```

However, the application opened the monitored data set first after the query command was issued the first time. The second query command shows you that some records have been read and passed since the last monitoring interval. The last query command shows you that no more data was processed and that the job has ended. The monitoring of the job is automatically stopped after the interval has expired twice.

```

:
+ IPSFP   INGY1305I Job Log Monitoring has been started for
          HIR-HIRZJLMS-JOB09337-AAAZOUT.
:
" IPSFP
Status of task INGTJLM: ACTIVE
Owner    Jobname  Jobid   DDname   Status   Freq. Last      Read  Passed
-----
HIR      HIRZJLMS JOB09337 AAZOUT  -----M-I 0:10   n/a      0     0
                                     0     0
*** Status complete ***
:
" IPSFP
Status of task INGTJLM: ACTIVE
Owner    Jobname  Jobid   DDname   Status   Freq. Last      Read  Passed
-----
HIR      HIRZJLMS JOB09337 AAZOUT  -A0--M-- 0:10  10:38:20    3     2
                                     3     2
*** Status complete ***
:
" IPSFP
Status of task INGTJLM: ACTIVE
Owner    Jobname  Jobid   DDname   Status   Freq. Last      Read  Passed
-----
HIR      HIRZJLMS JOB09337 AAZOUT  -A0--MJ- 0:10  10:38:51    3     2
                                     0     0
*** Status complete ***
:
+ IPSFP   INGY1306I Job Log Monitoring has been stopped for
          HIR-HIRZJLMS-JOB09337-AAAZOUT.
-----

```

Figure 5. Monitoring a non-SA z/OS controlled job.. For a detailed description of the status fields refer to "Status Information".

INGLINK

Purpose

The INGLINK command lets you:

- Activate and deactivate a link between a consumer and a provider application that is defined as a dynamic link in the automation policy
- Query the status of a link between a consumer and a provider application as defined in the automation policy

A *consumer* application executes commands based on the UP_, DN_, ISUP_, or ISDN_ prefixed messages defined in the MESSAGES/DATA policy item for the application. These messages contain pre-defined commands to be taken by the consumer application depending on the status of the provider application.

A *provider* application and its subsystem name form part of the UP_, DN_, ISUP_, or ISDN_ message suffix. These messages notify the consumer application of the provider status and trigger the predefined commands to be executed by the consumer application.

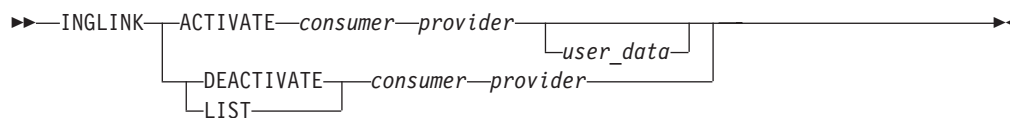
The messages defined in the consumer application MESSAGE/DATA policy item are as follows (subsys is the provider subsystem name):

- UP_subsys - actions taken by the consumer, when the provider changes to an UP status,
- DN_subsys - actions taken by the consumer, when the provider changes to DOWN status,

- ISUP_subsys - actions taken by the consumer, when it comes UP and the provider status is UP,
- ISDN_subsys - actions taken by the consumer, when it comes UP and the provider status is DOWN.

Note that there are no messages for when the consumer application goes down.

Syntax



Parameters

A(CTIVATE)

Activates a link between a consumer and a provider application that is defined as a dynamic link in the automation policy. The link state changes to ACTIVE.

This function requires a fully-qualified consumer and provider name, that is, the name cannot contain any wildcards. You do not need to specify the full resource name.

Subsystem or job names are accepted. Subsystem names have preference.

D(EACTIVATE)

Deactivates a link between a consumer and a provider application that is defined as a dynamic link in the automation policy. The link state changes to INACTIVE.

This function requires a fully-qualified consumer name. The provider name can be either fully-qualified (that is, the name cannot contain any wildcards) or an asterisk '*' to refer to all providers.

Subsystem or job names are accepted. Subsystem names have preference.

L(IST)

Displays links between consumer and provider applications and their current state.

This function supports wildcard names for consumers and providers. The consumer and provider names must be subsystem names.

consumer

The subsystem name of an application that uses services provided by another application.

provider

The subsystem name of an application that provides services used by another application.

user_data

User data can be set when you use the ACTIVATE function. The data is stored as additional information that is related to the activated link.

The data is provided in variables &EHKVAR1 through &EHKVAR9 and &EHKVART when a provider status change is processed by the ACTIVMSG or TERMMSG command. These variables can be used when defining commands for the UP_provider, DN_provider, ISDN_provider, ISUP_provider message ID of the consumer.

Return Codes

- 0 Normal completion.
- 4 Incorrect parameter.
- 5 Other error.
- 6 Initialization of the automation environment is not complete.
- 8 The link is not defined in the policy.

Restrictions and Limitations

Static and dynamic links can be defined in the policy between consumer and provider applications that are running on the same system.

The ACTIVATE function requires fully-qualified consumer and provider names, that is, the names cannot contain any wildcards.

The DEACTIVATE function requires a fully-qualified consumer name (that is, the name cannot contain any wildcards) and a fully-qualified provider name or an asterisk to refer to all providers.

Instead of using subsystem names, job names can also be used for the ACTIVATE and DEACTIVATE actions. INGLINK looks for a matching subsystem name first. If no matching subsystem name is found, it looks for a job name.

The LIST function allows wildcard names as consumer or provider names. Job names cannot be used.

INGLINK maintains the status of links based on common global variables. Access to these variables is made through the consumer's work operator in order to guarantee serialized exclusive access.

Usage

Static and dynamic links can be defined in the SA z/OS customization dialog. There you can define pseudo-messages in the MESSAGES/USER DATA policy of the consumer application with the prefix UP_ , DN_ , ISUP_ , or ISDN_ followed by the provider subsystem name (for example, UP_TCPIP). You can define a command action for these messages that is executed when the provider enters an UP or DOWN state (UP_ , DN_) or if the consumer application comes up and the provider is in an UP or DOWN state (ISUP_ , ISDN_). No additional definitions are required for the provider application.

For static links, no further definitions are required for the consumer application. Using INGLINK LIST you can list any static or dynamic links. Static links are always active links. Thus INGLINK ACTIVATE or INGLINK DEACTIVATE cannot be used for static links.

For dynamic links, a USER action must be applied to the pseudo-messages in the MESSAGES/USER DATA policy by setting the keyword to DYNAMIC and the value to YES. Dynamic links are initially inactive. These links can be activated with INGLINK ACTIVATE and deactivated with INGLINK DEACTIVATE. Dynamic links have the advantage that they can be activated and deactivated at run time. This may be necessary if a consumer requires services that can only be determined at run time.

Typically a user routine is required to identify the current provider. After the provider has been determined, INGLINK ACTIVATE can be called from the user

routine to activate the link to that provider. A configured action can be taken by the consumer if a provider starts or terminates only when a link is active.

The user routine is typically started along with the consumer application.

Examples

Example 1

Activating a dynamic link that is in an 'inactive' state:

```
INGLINK ACT MQ TCPIP1 optional user data
```

Console Output: ING004I REQUEST >INGLINK ACTIVATE< SUBMITTED

Log Output: A0F367I LINK ACTIVATED BETWEEN MQ AND TCPIP1

Example 2

Deactivating a dynamic link that is in an 'active' state:

```
INGLINK DEACT MQ TCPIP1
```

Console Output: ING004I REQUEST >INGLINK DEACTIVATE< SUBMITTED

Log Output: A0F368I LINK DEACTIVATED BETWEEN MQ AND TCPIP1

Example 3

You can list all links with the following command:

```
INGLINK LIST * *
```

The output lists all dynamic and static links that are defined in the automation policy.

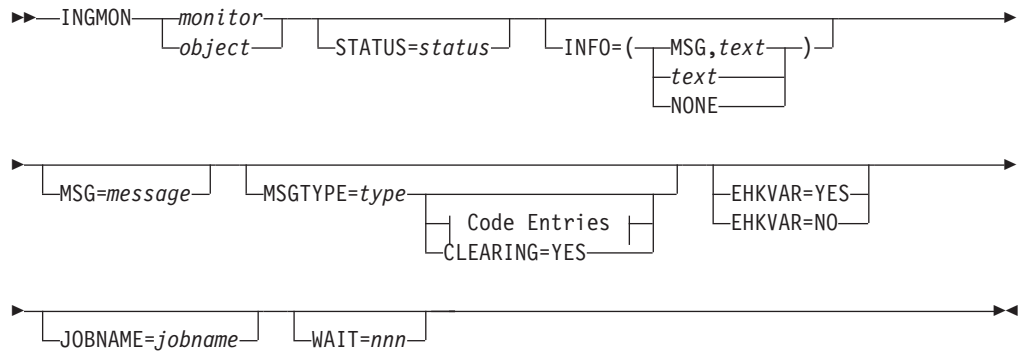
```
NV54 SA34 NM          Tivoli NetView IPUFA NETOP1 12/20/10 14:06:48
! IPUFA
Consumer      Provider      UP DN ISUP ISDN User Parameters
-----
CS1TRANS      PS1TRANS      - - S S
CS2            PS2A          - - S S
CS2            PS2B          - - S S
CS2            PS2C          - - S S
CS2A          PS2           - - S S
CS2B          PS2           - - S S
CS2C          PS2           - - S S
CS2TRANS      PS2TRANS      - - S S
CS3            PCS3          - - S S
CS4            CS4           - - S S
CS5            PS5           - - S S
C1            P1REN         - - S S
GFG2          PD2           S S I I
PCD3          PD3           - - I I
PCS3          PS3           - - S S
-----
CS1            PS1           - - S S
CS10THER      PS10THER      - - S S
```

INGMON

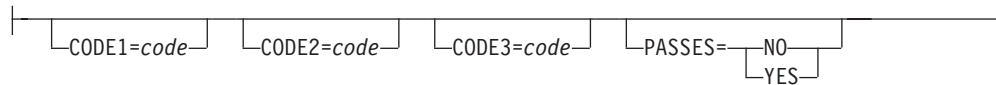
Purpose

INGMON is a command that can be called from the NetView automation table. You use it to inform SA z/OS of the status of a monitoring resource and to issue commands in response to a message or an OMEGAMON® exception.

Syntax



Code Entries:



Parameters

monitor

This is the name of a monitor resource. It can be specified in automation manager notation (for example, SAPMON/MTR/AOC8) or in agent notation.

Note: If you specify *monitor* in automation manager notation (that is, *xxx/MTR/yyy*) and such a resource does not exist on the local system, no search for a monitored object is performed. An error message is issued instead.

object

This is the name of a monitored object as defined in the MTR policy in the customization dialog. INGMON automatically finds the corresponding monitor resources and operates on them.

Note that a monitor resource with that name is searched first. Then, if it is not found, a search for a monitored object is performed.

Notes:

1. Be careful not to specify objects that have the same name as an existing monitor resource, otherwise INGMON will only find the entry for the monitor resource and not for the object. For example, if you have a monitor resource, ABC, and you define a second monitor resource, XYZ, with a monitored object called ABC, the call INGMON ABC will always find the monitor resource ABC and never the object ABC that has been defined for XYZ.
2. If you specify *object* in automation manager notation (that is, *xxx/MTR/yyy*) and such a resource does not exist on the local system, no search for a monitored object is performed. An error message is issued instead.

STATUS

This is the new state that the monitor has determined. The state represents

either the health status of the objects that the monitor is watching, or the state that the monitor is in. The latter can be one of the following:

FAILED

The monitor has failed. Recovery may be in progress. No acceptable health status was provided.

BROKEN

Both the monitor and recovery failed. This is a permanent condition. The monitor will not be re-invoked.

The health status of the object, or objects, that the monitor is watching are as follows, from the least to the most serious:

UNKNOWN

The health status is not yet available.

NORMAL

The monitor has obtained good results from the object, or objects, that it is watching.

WARNING

The monitor detected a certain degree of degradation in the operation of the monitored object.

MINOR

The same as WARNING, but more severe.

CRITICAL

The same as MINOR, but more severe.

FATAL

The same as CRITICAL, but more severe.

INFO

This defines the message that is associated with the new health status. The parameter value must be enclosed in parentheses, or single or double quotation marks. If not present, the value of the MSG parameter will be used instead. If neither parameter is present, text will be constructed from whatever is in the default safe.

MSG, *text*

The specified text is associated with the new health status.

In SDF, when displaying the new health status via message AOF550I, the text is shortened to "MESSAGE *message_ID* RECEIVED", where *message_ID* is the first token of the given text.

text

The specified text is associated with the new health status.

NONE

No message is associated with the status.

MSG

This defines the text of the message that commands or replies are defined for in the MESSAGES/USER DATA policy item of the policy database.

message

The message must be enclosed in parentheses, or single or double quotation marks. If not present, the content of the default safe will be taken instead.

MSGTYPE

This is the value entered in the **Message ID** field in the customization dialog for the MESSAGES/USER DATA policy item (the entry type field in the automation control file entry for the command). MSGTYPE is typically coded with the message ID or the OMEGAMON exception identifier, such as XCHN or SWPC. If not present, the first token of the value of MSG parameter will be used instead. If neither parameter is present, the message ID will be extracted from the message in the default safe.

The task global variables EHKVAR n contain the parsed message from either the MSG parameter or, if not present, the message in the default safe.

CODE n

When specified, the passed codes are used to search the code entries for a particular Message ID or exception that is specified in the policy item MESSAGES/USER DATA. The first token of the value returned is used as an option to select the commands to be issued from the automation control file (the value gives a set of commands). If no match occurs for the specified codes, or if no codes are specified, the value ALWAYS is used to select the commands to be issued.

As a selection you must specify one of the following:

Perform pass processing and execute all commands with a selection matching the current pass or blank.

Note that this overrides the PASSES keyword with YES.

#type Interpret *type* as a message type. Perform pass processing for this message and execute all commands with a selection matching the current pass or blank.

Note that this overrides the PASSES keyword with YES.

This is like calling INGMON with MSGTYPE=*type*.

Others

Execute all commands with the given selection, or blank.

The second token of the value returned is used to determine the health status to be set. If specified it must denote one of the values listed under the STATUS keyword. If no match occurs, or the second token is omitted, the value given on the STATUS keyword is used.

The CODE parameters are mutually exclusive to the PASSES=YES parameter.

PASSES

Specifies whether passes are used to issue the commands. The INGMON command interrogates the automation control file to see if passes are specified in the command entries. If so, PASSES=YES is defaulted unless PASSES=NO was specified when calling INGMON.

NO Passes are not used to issue the commands.

YES

Passes are used to issue the commands. The pass count is incremented every time INGMON is called. The pass count is keyed by monitor name and by message type or exception (that is, the OM exception). The count is automatically reset when the monitor resource is deactivated, or when INGMON is invoked with the CLEARING option.

CLEARING

Indicates that this is a clearing event. The situation that caused the message or

exception is no longer present. It resets the pass count and removes the mask (DISABLETIME) for this message or exception.

EHKVAR

This parameter determines whether the tokens of the parsed message text are to be stored in task global variables EHKVAR0 through EHKVAR9 and EHKVART.

YES

The tokens of the triggering message are to be assigned to the task global variables EHKVAR n .

NO No values are to be assigned to the task global variables EHKVAR n .

JOBNAME

Indicates that the job name that is given is to be used rather than the one obtained by the jobname() function to match a monitored object to the corresponding monitor resources.

If the monitor resource contains a value in the monitored job name field it is only considered a match when both the monitored object and the given job name match.

If JOBNAME is omitted the jobname() function is used to determine the name of the job that issued the triggering message. If a job name cannot be determined the value N/A is used.

JOBNAME is required when INGMON is not message driven.

WAIT

Specifies the number of seconds to wait before reporting that a timeout occurred if the automation manager does not provide the requested data. The maximum time interval is 999 seconds.

If omitted, the time interval is 30 seconds.

Return Codes

- 0 Okay.
- 1 An error occurred.
- 2 A monitor resource or monitored object with the specified name does not exist.

Restrictions and Limitations

Status cannot be changed via STATUS=option while the monitor is in status BROKEN. Consider the RESET=YES option.

Usage

You should normally call the INGMON command from the NetView automation table.

For more details about the task global variables that can be used with INGMON see "Task Global Variables" in *IBM Tivoli System Automation for z/OS Customizing and Programming*.

The message that caused the INGMON call is stored in the SAFE named AOFMSAFE. All commands and replies that are triggered through INGMON have access to this SAFE.

Examples

Example 1

This example demonstrates a call pager routine when channel path 26 is not operational. The MESSAGE/USER DATA policy definition contains an entry for "+XCHN" as follows:

CMD entry

```
PAGER &SUBSAPPL,&EHKVAR0,&EHKVAR4
```

Where:

- PAGER is the name of the clist that handles the paging
- &SUBSAPPL contains the monitor name
- &EHKVAR0 contains the exception ID
- &EHKVAR4 contains the CHPID

Note that the "+ " prefix, written as a '+' followed by a blank in front of the exception identifier, is used to distinguish a normal message from an OMEGAMON exception.

CODE entry

Code 1	Code 2	Code 3	Value Returned
26	*		ALWAYS
*	*		IGNORE

OVR entry

The standard AT entry pattern is generated by SA z/OS. You can change the condition statement to assign TOKEN(10), which contains the missing channel number to a variable, for example, MISSCHAN.

The pattern of the action statement would be changed to pass that variable to INGMON, that is:

```
EXEC (CMD('INGMON monitor CODE1='MISSCHAN))
```

It is assumed that token 10 in message ING080I contains the channel path. The command fragment up to the monitor name is automatically generated by the customization dialog. The message type (that is, exception ID) is available to INGMON indirectly through the default SAFE.

You can append additional parameters through such an OVR entry.

Example 2

Suppose the installation fixed the problem with the missing channel path as shown in the previous example. To indicate that the situation that caused the exception no longer exists and that the recovery action has been successfully processed, use:

```
INGMON monitor MSGTYPE=XCHN CLEARING=YES
```

Example 3

This example shows how to send an alert when CICS Link LNKA2B is not operational. The MONITOR INFO policy definition contains the following information:

Monitored Object

```
CPSM.CICSA.CONNECT.LNKA2B
```

Inform List

```
IOM
```

The MESSAGE/USER DATA policy definition contains an entry for ING150I as follows:

CODE entry

Code 1	Code 2	Code 3	Value Returned
L*	*	*	NOP WARNING
HW	*	*	NOP MINOR
*HS	*	*	#MSGCRIT CRITICAL
*	*	*	NOP NORMAL

Because no commands should be issued for the health statuses WARNING, MINOR, and NORMAL, use an arbitrary selection (NOP). A selection is required because the health status must be the second token.

For the CRITICAL case we have INGMON jump to message type MSGCRIT and execute the commands listed there. In our example this would probably be an INGALERT command.

We want the pass counts to be reset as soon as the health status returns to NORMAL, so we specify in the HEALTHSTATE policy definition:

```
INGMON monitor MSGTYPE=MSGCRIT CLEARING=YES
```

INGMTRAP

Purpose

The INGMTRAP command facilitates the use of INGOMX for OMEGAMON exception monitoring using Monitor Resources. It traps one or more OMEGAMON exceptions of interest and generates ING080I messages exposed to automation for each exception that is found, in order to set the Monitor Resource's health state and to issue commands to react to such exceptional conditions.

Syntax

```
→ INGMTRAP NAME=session_name XTYPE=(exception) →
```

Parameters

NAME=*session_name*

This is the name of the OMEGAMON session as defined in the automation policy that exceptions should be monitored from.

XTYPE=*exception_list* | *exception*

A list of one or more OMEGAMON exceptions. If the list consists of more than one exception, it must be put in quotation marks or parentheses, and either commas or blanks must be used to separate the exceptions.

Return Codes

1 BROKEN

INGMTRAP failed to communicate with the OMEGAMON monitor denoted by the *session_name*. Either SA z/OS was unable to create the session, or the session was prematurely terminated for other reasons and could not be

INGMTRAP

re-established. In general, whenever the state of a session is neither INACTIVE nor ACTIVE, the health status BROKEN indicates that operator intervention is required.

2 FAILED

INGOMX detected that the session denoted by *session_name* exists but no output was received by OMEGAMON, for example due to a timeout. This situation may go before the monitor runs the next time. In general, whenever the state of a session is ACTIVE, the health status FAILED indicates a temporary problem that is likely solved the next time the monitor runs.

3 NORMAL

No exceptions have tripped. The health state of this Monitor Resource is therefore normal. Message ING081I was built and added to the Monitor Resource's history to indicate that no exception was found. ING081I will not be exposed to automation.

8 DEFER

Exceptions have tripped and message ING080I was built for each. The health state of the Monitor Resource will be set on behalf of message automation for ING080I when the message is processed by the automation table. An existing health state remains in effect until a new health state is set during message automation.

Usage

To monitor OMEGAMON exceptions with a Monitor Resource, specify INGMTRAP as the monitor command in the customization dialog when defining the Monitor Resource. INGMTRAP will generate an ING080I message for each exception that tripped and that was specified with XTYPE and exposes the message to automation.

In order to react to such exceptions, use the MESSAGES/USER DATA policy item for that Monitor Resource to specify the health status and optionally the recovery activities for each exception. As indicated by the return code DEFER, the health status of the Monitor Resource will only be updated if an ING080I message was correctly processed.

Example

To monitor the existence of a LOGN exception that is issued by OMEGAMON for DB2 whenever the number of primary active logs falls below the installation-specified threshold, enter the following as a monitor command in the MTR policy:

```
INGMTRAP NAME=OMSY4DB XTYPE=LOGN
```

INGOMX

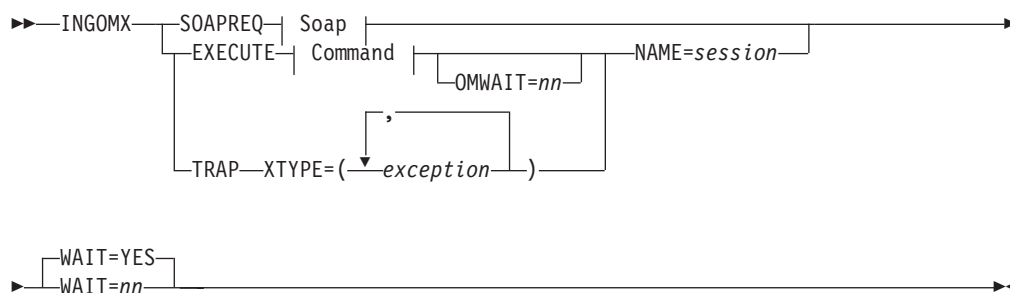
Purpose

INGOMX is a programming interface that provides interaction capabilities with OMEGAMON. It allows a program or command list to invoke OMEGAMON exception analysis in order to trap one or more exceptions of interest or to issue one or more OMEGAMON commands. The response generated by OMEGAMON on behalf of a request is written to the console but not exposed to automation.

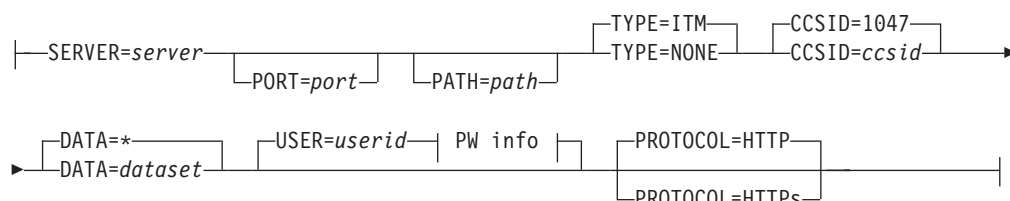
Additionally, it provides an interface to communicate with an IBM Tivoli Monitoring SOAP server to issue SOAP messages and to process the response from

the SOAP server. While INGOMX handles the communication and the envelope of the SOAP message, it is the responsibility of the caller to provide an appropriate body either dynamically or in a data set of choice. The body consists of specific elements in XML notation that denote the particular request, the target application for this request, user ID and password, and other request-specific information.

Syntax



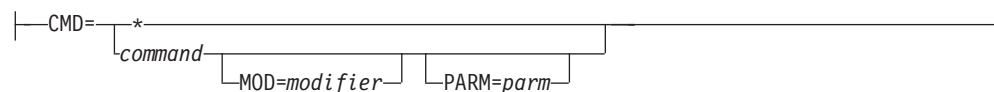
Soap:



PW info:



Command:



Parameters

ATTACHSESSION

This is the function code to request that an OMEGAMON session with a named set of attributes is established and a user (according to these attributes) is logged on.

DETACHSESSION

This is the function code to request that the user that is currently logged on to the OMEGAMON session is logged off and that the OMEGAMON session specified by *session_name* is destroyed.

TERMSESSION

This is the function code to end the OMEGAMON session. The function is similar to the DETACHSESSION request but, in addition, the session is locked

and ATTACHSESSION is not allowed. This function code is used only by SA z/OS initialization and only SA z/OS initialization can unlock the session.

SOAPREQ

The function code to send a SOAP message to the SOAP server designated by *server*. The SOAP message to be sent can be built dynamically and passed to INGOMX through the default safe or it can be in a user-defined data set.

SERVER=*server*

server refers to the SOAP server defined in the SOAP SERVER policy of the NETWORK (NTW) entry in the automation policy.

Alternatively, *server* can be the IP-address (if it starts with a digit or contains ':') or the symbolic host name. If a SOAP SERVER policy is found for *server*, it will be used preferably.

PORT=*port*

port refers to the port number that the SOAP server is listening to. This is optional and only required if *server* does not denote a definition in the SOAP SERVER policy.

PATH=*path*

path refers to the absolute path that, together with the host address and the port number, forms the address of the SOAP service. An absolute path must start with a slash ('/'). This is optional and only required if *server* does not denote a definition in the SOAP SERVER policy.

TYPE={ITM|NONE}

The type refers to the kind of SOAP request that is being issued. For ITM type SOAP requests, the response document is transformed into messages that can be processed further using NetView PIPEs. For other SOAP requests, the response is returned in XML format instead.

CCSID=*ccsid*

ccsid refers to the character set identifier used to encode the XML source and target documents. The SOAP request is translated from *ccsid* to UTF-8 before it is sent to the SOAP server and it is translated back to *ccsid* upon receipt.

DATA={*|*dataset*}

If * is specified, the SOAP message is located in the default SAFE. This assumes that the caller has either created the SOAP message dynamically or obtained it from some other source and passed it to INGOMX via a PIPE input stream. See also "Example 3: Send SOAP Message Using the Default Safe" on page 116.

dataset refers to the name of either a sequential data set or a partitioned data set, including the member name where INGOMX finds the SOAP message.

See paragraph "Using INGOMX Directives" on page 112, "Example 8: Using INGOMX Directives for defining SOAP Data" on page 119, and "Example 9: Using Ingomx_SOAP_Envelope Directive for SOAP12" on page 119 for the usage of the alternate definition of SOAP Body, SOAP Envelope, and HTTP header records using directives.

USERID=*userid*

The userid required for authentication to the SOAP server, if security is enabled at the SOAP server.

PASSWORD=SAFPW|*password*

The password required for authentication, specified either in plain text, or

deposited in NetView's Password Data Set under the domain name SOAP. In this case, the real password must be specified under NetView using the following NetView command:

```
GETPW userid SOAP,INIT=password
```

PROTOCOL

The protocol for the socket connection type. TLS/SSL security is selected with HTTPS, and the default is HTTP. It is flagged as an error if you define it in addition to a server object.

EXECUTE

The function code to issue a command to the OMEGAMON session that is specified by *session_name*. The command and its parameters are specified by the CMD, MOD, and PARM keywords. The output of this command corresponds to the output produced by the OMEGAMON monitor on a 3270 screen.

CMD={*command* | *}

This is the pure 1 to 4 character OMEGAMON command that is issued on the OMEGAMON session without parameters.

If an asterisk (*) is passed instead of a command, INGOMX expects a list of up to 22 OMEGAMON commands in the default SAFE. This allows a programmer to issue multiple commands at once, in particular, minor commands that require the presence of an appropriate major command.

When commands are passed within the default SAFE, the same syntax rules apply as denoted in the syntax diagram above, that is, each command is specified with the CMD keyword followed by an optional modifier and optional parameter string.

The command that is specified or the first command in the default SAFE will be logged in the NetView log with message ING083I.

MOD=*modifier*

modifier is an optional additional character that will be inserted by SA z/OS in front of a command to modify the behavior of that command. For example, a '<' modifies the command ALLJ such that instead of one line of address spaces, all address spaces are returned at once. Refer to the OMEGAMON command manual for a reference of modifiers allowed for each particular command. The default modifier is a blank character.

PARM=*parm*

parm is an optional parameter string of up to 74 characters. The parameter string will be appended by SA z/OS to the command, if specified. The default parameter is a NULL-string.

OMWAIT=*nn*

Several OMEGAMON commands begin a process that accumulates data over a small period of time. For example, the OMEGAMON for MVS command MCPU accumulates data about CPU status, and then displays this data. To mimic this functionality, OMWAIT can be used, where *nn* is the optional time during which data is accumulated before the command is issued again, and can be between 0 and 59 seconds. The default wait time is 0.

TRAP

This is the function code to filter exceptions that are reported by the OMEGAMON session specified by *session*. The exceptions that should be filtered are specified by the XTYPE keyword. The output of this command

corresponds to filtered output produced by the OMEGAMON monitor exception analysis on a 3270 screen, where only the selected exceptions are included.

XTYPE={*exception_list*|*exception*}

A list of one or more OMEGAMON exceptions. If the list consists of more than one exception, it must be put in quotation marks or parentheses, and either commas or blanks used to separate the exceptions.

NAME=*session*

This is the name of the OMEGAMON session as defined in the automation policy.

WAIT={**YES**|*nnn*}

This parameter determines the interval after which a request is terminated with a timeout condition.

If WAIT=YES is specified (default), either the default wait time is used as a timeout value (WAITTIME) or, for OMEGAMON sessions, the timeout that is specified in the OMEGAMON SESSION policy. The parameter default variable INGOMX_WAIT can be used to override the default value in WAITTIME.

If WAIT=*nnn* is specified, *nnn* denotes a positive number in units of seconds.

Using INGOMX Directives

The SOAP message specified in parameter DATA is a section of XML used as a SOAP body, which is enclosed in a SOAP 1.1 Envelope and prefixed with HTTP 1.1 records. If these data are not sufficient for the requested webservice, for example, if there is a missing SOAPAction, it is possible to redefine or define additional SOAP and HTTP fields using the alternative INGOMX directives.

If the SOAP message in DATA starts with a Ingomx_Directives tag, the whole DATA parameter is expected to be a Directive XML using the following tags to describe the user data:

<Ingomx_SoapAction>

this field replaces the default SOAPAction field.

<Ingomx_SOAP_Body>

this field contains the SOAP Body parameter to INGOMX. If this directive is specified, the default SOAP envelope is added.

<Ingomx_SOAP_Envelope>

this field contains the complete SOAP message, for example, the SOAP envelope and the SOAP Body WITHOUT the one line XML declaration. The default XML declaration `<?xml version="1.0" encoding="utf-8"?>` will be inserted by INGOMX automatically. Use this tag, when you need full control of the SOAP envelope content.

Note: Ingomx_SOAP_Envelope and Ingomx_SOAP_Body tags are mutually exclusive.

<Ingomx_ContentType>

this field replaces the default Content-Type record.

<Ingomx_Host>

this field replaces the default HTTP HOST header record.

<Ingomx_POST>

this field even replaces the default HTTP POST request record.

<Ingomx_HTTP_Addon>

this field may specify an additional HTTP header record.

If a tag is specified without data, like <Ingomx_Host/>, the whole record is removed from the SOAP request.

Return Codes

- 0 Normal completion. The filtered output of the exception analysis or the response of the OMEGAMON command or the SOAP request, respectively, is written to the console.
- 3 The operator invoking INGOMX is not authorized to issue this request for any of the following reasons:
 - The caller is not allowed to access the session indicated by *session_name*
 - The caller is not allowed to issue the OMEGAMON command (or commands) specified by CMD
 - The caller is not allowed to send the SOAP message to the SOAP server indicated by *server_name*

Update the NetView command authorization table or the RACF definitions, or both, for the named session, the named SOAP server, and command (or commands).

NetView issues BNH236E and BNH237E with detailed error information.

- 1 INGOMX failed to communicate with the session whose *session_name* was passed as input. The *session_name* is unknown or does not refer to a valid OMEGAMON session. In case of an invalid OMEGAMON session, message ING084I is written to the netlog.

If the target was a SOAP server, the *server_name* is unknown or does not refer to a host that can be reached through the IP network. Refer to message ING164I for further diagnostic details in the netlog.
- 3 An internal error occurred. Message ING084I is written to the netlog and provides more detailed error information.
- 4 Syntax error. Invalid parameters were passed to INGOMX. Refer to the netlog for additional error information.
- 5 Timeout occurred. The requested operation was interrupted due to a timeout as specified for this session in the customization dialog. The timeout value might be too low or more session operators might be required.
- 6 The command environment for INGOMX was not appropriately initialized at the time this command was issued. Possible reasons are that the agent is currently being initialized or a cold start of the automation control file is being done.
- 7 Creation of the NetView Terminal Access Facility (TAF) session failed or VTAM is not available. Message ING084I is written to the netlog and provides more detailed error information.
- 8 The user ID that is specified in the session definition for *session_name* cannot log on to OMEGAMON. Session creation failed.
- 9 The requested function is not allowed within the current session state. Refer to the INGSESS command (in *IBM Tivoli System Automation for z/OS Operator's Commands*) for the current state and available options.
- 10 A SOAP request could be delivered, but the SOAP server failed to interpret the request correctly. Messages ING162I and ING163I are returned to the caller for further information about this failure.
- 11 A SOAP request could not be delivered to the SOAP server because the service has been moved to a new location. Message ING167I is returned to the caller indicating the new location that must be used instead.

- 12 The specified data set containing the SOAP request could not be allocated or read. Message ING164I is written to the netlog indicating the code returned from PIPE QSAM.
- 13 The specified CCSID does not exist. Specify a valid CCSID.

Usage

Invoke INGOMX in a PIPE to capture and further analyze the output produced by OMEGAMON. Alternatively, INGOMX can also be called directly from the operator console, but note that INGOMX does not issue any confirmation messages and that output is only written to the console when INGOMX returned with code 0.

Refer to "Controlling Access to IBM Tivoli Monitoring Products" in *IBM Tivoli System Automation for z/OS Planning and Installation* for Command Authorization Table identifiers supported by INGOMX.

Examples

Example 1: Use of INGOMX in a User-Written Monitor Command Routine

The following command list is specified as the monitor command that is periodically invoked to find out the usage of CSA and SQA below 16MB monitored by OMEGAMON for MVS. The session was defined under the session name OMSY4MVS. The OMEGAMON command for this purpose is "CSAA." The monitor command will return the following health states:

- NORMAL (3) when the utilization of both CSA and SQA is below 50%
- WARNING (4) when the utilization is below 70%
- MINOR (5) if below 80%
- CRITICAL (6) if below 90%
- FATAL (7) otherwise

```

/*-----*/
/* Constants and variables */
/*-----*/
Rc_Unknown = 0
Rc_Failed = 2
Rc_Normal = 3
Rc_Warning = 4
Rc_Minor = 5
Rc_Critical = 6
Rc_Fatal = 7

health_state = Rc_Unknown
lrc = 0

ss='ff'x
ec='fe'x
dc='fd'x
/*-----*/
/* Use PIPE to call INGOMX */
/*-----*/
"PIPE (STAGESEP "||ss||" END "||ec||" NAME API)",
  " NETV (MOE) INGOMX EX,NAME=OMSY4MVS,CMD=CSAA",
  ss|"A: LOCATE 1.8 /DW0369I /",
  ss|" VAR dwo369",
  ec|"A: ",
  ss|" STEM output."

/*-----*/

```

```

/* Monitor failed if INGOMX return code was not zero */
/*-----*/
if symbol('dwo369') = 'VAR' then
  do
    parse var dwo369 . 'RETURN CODE' lrc '.'
    monmsg = 'INGOMX return code was RC='lrc+0
    health_state = Rc_Failed
  end
else
/*-----*/
/* Filter CSA and SQA percentages to derive health state */
/*-----*/
do
  csa = 0; sqa = 0
  do i=1 to output.0
    if pos('+ CSA',output.i) > 0 then
      parse var output.i . 'CSA' . . . . csa '%' .
    if pos('+ SQA',output.i) > 0 then
      parse var output.i . 'SQA' . . . . sqa '%' .
    end i
  select
    when csa < 50 & sqa < 50 then health_state = Rc_Normal
    when csa < 70 & sqa < 70 then health_state = Rc_Warning
    when csa < 80 & sqa < 80 then health_state = Rc_Minor
    when csa < 90 & sqa < 90 then health_state = Rc_Critical
    otherwise
      health_state = Rc_Fatal
  end
  monmsg = 'CSA usage is 'csa+0'%, SQA usage is 'sqa+0'%.'
end

'PIPE VAR monmsg | CONSOLE ONLY'

```

Return health_state

Example 2: Use of INGOMX to Display Jobs of Interest

The following command list can be entered from the console to show a list of jobs that have fixed storage occupancy greater than 1 MB. The command list traps the FXFR exception that is reported by OMEGAMON for MVS with the session name OMSY4MVS. From the list of exception lines that is returned by OMEGAMON, the command list selects those jobs that obtain more than 1 MB of fixed storage.

```

/*-----*/
/* Constants and variables */
/*-----*/
OneMB = 1024 * 1024
fframes = 0 /* Number of fixed frames */
fstor = 0 /* Fixed storage [Bytes] */
out. = 0 /* Jobs with >1MB fixed frames */
j = 0 /* Miscellaneous counter */
lrc = 0 /* Local return code */
ss = 'ff'x
ec = 'fe'x

/*-----*/
/* Use PIPE to call INGOMX */
/*-----*/
"PIPE (STAGESEP "||ss||" END "||ec||" NAME API)",
" NETV (MOE) INGOMX TRAP,NAME=OMSY4MVS,XTYPE=FXFR",
ss|"A: LOCATE 1.8 /DWO369I /",
ss|" VAR dwo369",
ec|"A: ",
ss|" STEM output."
/*-----*/
/* Command failed if INGOMX return code was not zero */
/*-----*/

```



```

if symbol('dwo369') = 'VAR' then
  do
    parse var dwo369 . 'RETURN CODE' lrc '.'
    'MESSAGE DSI072 FXSHOW INGOMX 'lrc+0
  end
else
/*-----*/
/* Produce list of address spaces with >1MB fixed frames          */
/* + FXFR STC NETVBOW      | Fixed Frames in use = 414            */
/*-----*/
do
  out.1 = 'FXMON: Jobs with fixed storage > 1MB'
  out.0 = 1
  do i=2 to output.0
    parse var output.i '+ FXFR' . job . 'Frames in use = 'fframes .
    fstor = fframes * 4096
    if fstor > OneMB then
      do
        j = out.0 + 1
        out.j = left(job,8) format(fstor/OneMB,4,2)||'MB'
        out.0 = j
      end
    end i

    'PIPE STEM out. | CONSOLE ONLY'
  end

Exit 0

```

Example 3: Send SOAP Message Using the Default Safe

This example assumes that the SOAP server SOAPHUB was defined in the SA z/OS customization dialog SOAP SERVER policy. To request a list of address spaces starting with TEST and analyze their CPU utilization, you can construct and send a SOAP message like the one below to SOAPHUB in the following way:

```

smsg.1 = '<CT_Get>'
smsg.2 = ' <object>Address_Space_CPU_Utilization</object>'
smsg.3 = ' <attribute>job_name</attribute>'
smsg.4 = ' <attribute>asid</attribute>'
smsg.5 = ' <attribute>tcb_percent</attribute>'
smsg.6 = ' <afilter>Job_Name;LIKE;TEST*</afilter>'
smsg.7 = '</CT_Get>'
smsg.0 = 7

```

```

'PIPE (NAME SOAPREQ)',
'| STEM smgs.',
'| SAFE *',
'| NETV INGOMX SOAPREQ SERVER=SOAPHUB DATA=*'

```

Example 4: Send SOAP Message Using a Data Set

As an alternative, the XML source in “Example 3: Send SOAP Message Using the Default Safe” can also be stored in a data set. The invocation of INGOMX would then be:

```

'PIPE (END % NAME SOAPREQ)',
'| NETV (MOE) INGOMX SOAPREQ SERVER=SOAPHUB DATA=USER.SOAP.DATA(GET)',
'| L: LOC 1.8 'del||'DWO369I '||del,
'| EDIT SKIPTO 'del||'RETURN CODE' ||del,
'| UPTO 'del||'.' ||del,
'| WORD 3 1',
'| VAR my_retcode',
'| % L:',
'| CON ONLY'

```

The variable *del* is a delimiter character that does not appear in the data stream that is returned, for example, X'0D'.

Example 5: Explicit Specification of SOAP Server

Here an example is shown where the SOAP server is not defined in a SOAP server policy but directly in the invocation of INGOMX:

```
Server = 'boekeya.boeblingen.de.ibm.com'
Path = '///cms/soap'

Address Netvasis 'PIPE (END % NAME SOAPREQ)',
'| NETVASIS NETV (MOE) INGOMX SOAPREQ SERVER='Server,
'| PORT=1920 PATH=""Path" DATA=USER.SOAP.DATA(GET)',
'| L: LOC 1.8 'del||'DW0369I '||del,
'| EDIT SKIPTO 'del||'RETURN CODE' ||del,
'| UPTO 'del||'. ' ||del,
'| WORD 3 1',
'| VAR my_retcode',
'| % L:',
'| CON ONLY'
```

Note that the path is passed in double-quotation marks to preserve its case.

Example 6: Explicit Specification of SOAP Server, User ID, and Password

The following is an example where the SOAP server, user ID, and password are specified at the invocation of INGOMX:

```
Server = 'boekeya.boeblingen.de.ibm.com'
Userid = 'SoapUser'
Path = '///cms/soap'
Address Netvasis 'PIPE (END % NAME SOAPREQ)',
'| NETVASIS NETV (MOE) INGOMX SOAPREQ SERVER='Server,
'| USERID=""Userid" PASSWORD="SAFPW"',
'| PORT=1920 PATH=""Path" DATA=USER.SOAP.DATA(GET)',
'| L: LOC 1.8 'del||'DW0369I '||del,
'| EDIT SKIPTO 'del||'RETURN CODE' ||del,
'| UPTO 'del||'. ' ||del,
'| WORD 3 1',
'| VAR my_retcode',
'| % L:',
'| CON ONLY'
```

Note that the path, user ID and password are passed in double quotation marks to preserve their case. If the specified password is "SAFPW", the actual password is taken from the NetView Password Data Set. The password must be stored in the Password Data Set before it is used by using the following command:

```
GETPW SoapUser SOAP,INIT=password
```

Example 7: Secure Socket Connection

This example shows usage of secure port 3661 and prerequisites in z/OS.

```
Address Netvasis 'PIPE (END %NAME SOAPTLS)',
'|NETVASIS NETV (MOE)INGOMX SOAPREQ SERVER=<ip_addr of TEMS>',
'|PORT=3661 PATH=///cms/soap PROTOCOL=HTTPS DATA=USER.SOAP.DATA(GET)',
'|L:LOC 1.8 'del||'DW0369I' ||del,
'|EDIT SKIPTO 'del||'RETURN CODE' ||del,
'|UPTO'del||'. ' ||del,
'|WORD 3 1 ',
'|VAR my_retcode',
'|% L:',
'| CON ONLY'
```

The following steps are required to exploit secure sockets:

- Policy Agent Setup

Please refer to the *z/OS Communication Server* documentation for details. Be aware

that the TCP/IP profile has to contain the statement "TCPCONFIG TTLS" to result in the activation of the processed policy definitions.

- AT-TLS Policy

Figure 6 is a sample AT-TLS policy with the highest TCPIP trace. Please specify <tlsKeyring> and <ip_addr> accordingly.

```

TTLSRule                                NV_TEMS_WIN
{
  LocalAddr                              ALL
  RemoteAddrRef                          addr_TEMS
  LocalPortRange                          0
  RemotePortRange                        3661
  Direction                              Outbound
  Priority                                255
  TTLSGroupActionRef                     XXGRP
  TTLSEnvironmentActionRef               XXENV
  TTLSConnectionActionRef                XXCON
}
TTLSGroupAction                          XXGRP
{
  TTLSEnabled                            On
}
TTLSEnvironmentAction                    XXENV
{
  HandshakeRole                          Server
  EnvironmentUserInstance                 0
  TTLSKeyringParmsRef                    keyRing
  TTLSEnvironmentAdvancedParmsRef        XXADV
  Trace                                  255
}
TTLSConnectionAction                    XXCON
{
  HandshakeRole                          Client
  Trace                                  255
}
TTLSEnvironmentAdvancedParms             XXADV
{
  ApplicationControlled                   Off
  ClientAuthType                          PassThru
}
TTLSKeyringParms                         keyRing
{
  Keyring                                 <tlsKeyring>
}
IpAddr                                   addr_TEMS
{
  addr                                     <ip_addr>
}

```

Figure 6. Sample AT-TLS policy

- Certificate registration in keyring

The ITM Soap Server sends a self-signed certificate which has to be registered in the keyring. The certificate can be obtained easily if a webrequest is sent from a workstation browser. Use the following URL for this purpose:
https://<ip_addr>:3661//cms/soap/kshhsoap.htm.

You are asked to accept or deny the certificate. Store this certificate in X.509 PEM format (base64), upload this file to z/OS with Ascii to Ebcidic translation and add it to your keyring.

For RACF users, the following commands would complete the job:

```

raccert id(stcuser) add ('<UID.ITM.PEM>') WITHLABEL ('ITM') TRUST
raccert id(stcuser) addring(<keyring>)
raccert id(stcuser) connect (ID(stcuser) RING(<keyring>) LABEL('ITM') USAGE(CERTAUTH)
setropts raclist (digtring) refresh
setropts raclist (digtcert) refresh

```

Example 8: Using INGOMX Directives for defining SOAP Data

The following is an example that demonstrates the use of a SOAPAction header.

```

server = 'www.mywebservice.com'
port   = 80
path   = '/stockprice.asmx'
data   =,
  '<Ingomx_Directives>' ||,
  ,
  '<Ingomx_SoapAction>' ||,
  'SOAPAction: "http://www.mywebservice.com/GetPrice"' ||,
  '<Ingomx_SoapAction>' ||,
  ,
  '<Ingomx_SOAP_Body>' ||,
  ,
  '<GetPrice xmlns="http://www.mywebservice.com/">' ||,
  '<symbol>IBM</symbol>' ||,
  '</GetPrice>' ||,
  ,
  '<Ingomx_SOAP_Body>' ||,
  ,
  '</Ingomx_Directives>'

address NetVAsIs 'PIPE VAR DATA | COLLECT'
  '|NETV INGOMX SOAP SERVER='server 'PORT='port,
  '|PATH='path 'DATA=* TYPE=NONE PROTOCOL=HTTP',
  '|CONSOLE ONLY'

```

Example 9: Using Ingomx_SOAP_Envelope Directive for SOAP12

```

server = 'www.mywebservice.com'
port   = 80
path   = '/weather.asmx'
data   =,
  '<Ingomx_Directives>' ||,
  ,
  '<Ingomx_Host>' ||,
  'Host: www.mywebservice.com' ||,
  '<Ingomx_Host>' ||,
  ,
  '<Ingomx_SoapAction>' ||,
  'SOAPAction: "http://www.mywebservice.com/GetWeather"' ||,
  '</Ingomx_SoapAction>' ||,
  ,
  '<Ingomx_ContentType>' ||,
  'Content-Type: application/soap+xml; charset=utf-8' ||,
  '</Ingomx_ContentType>' ||,
  ,
  '<Ingomx_SOAP_Envelope>' ||,
  ,
  '<soap12:Envelope>' ||,
  'xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"' ||,
  'xmlns:xsd="http://www.w3.org/2001/XMLSchema"' ||,
  'xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">' ||,
  '<soap12:Body>' ||,
  '<GetWeather xmlns="http://www.mywebservice.com/">' ||,
  '<City>Stuttgart</City>' ||,
  '<Country>Germany</Country>' ||,
  '</GetWeather>' ||,
  '</soap12:Body>' ||,
  '</soap12:Envelope>' ||,
  ,

```

INGOMX

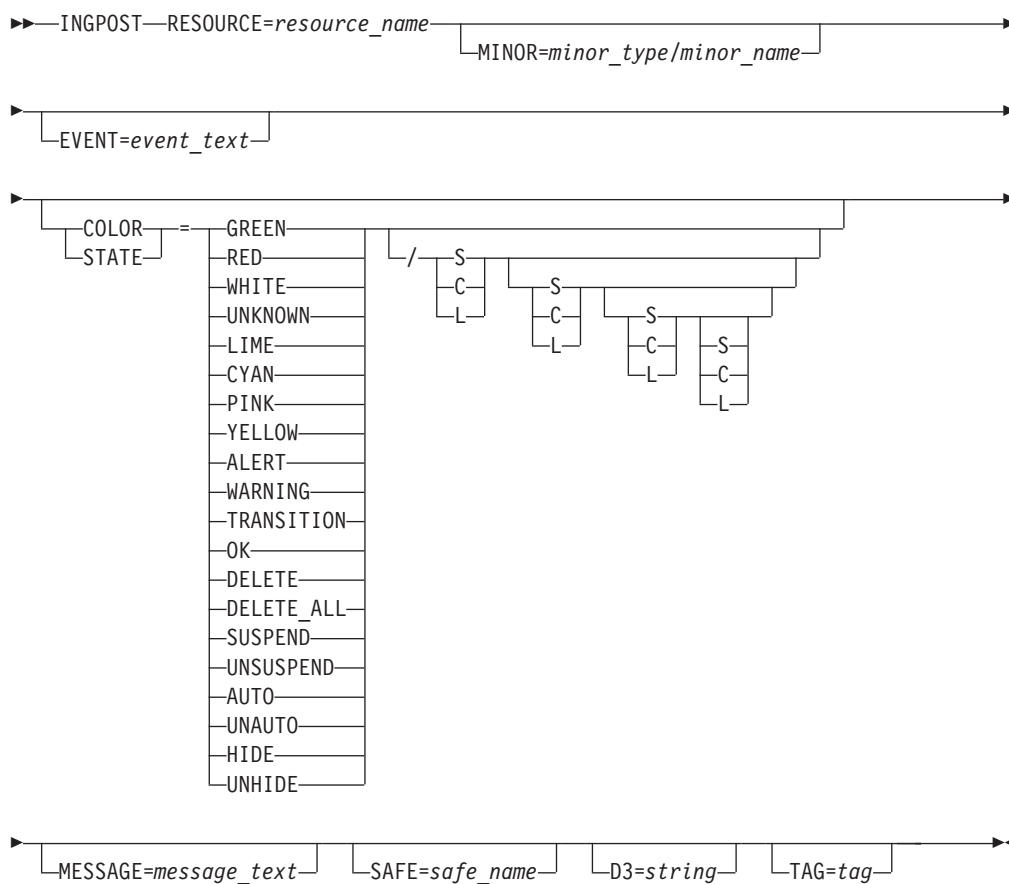
```
'</Ingomx_SOAP_Envelope>' ||,  
'</Ingomx_Directives>'  
  
address NetVAsIs 'PIPE VAR DATA | COLLECT'  
'|NETV INGOMX SOAP SERVER='server 'PORT='port,  
'|PATH='path 'DATA=* TYPE=NONE PROTOCOL=HTTP',  
'|CONSOLE ONLY'
```

INGPOST

Purpose

The INGPOST command posts status notifications to SA z/OS's NMC-based user interface.

Syntax



Parameters

RESOURCE

Specifies the name of the major resource that the notification is associated with.

resource_name

Is a standard SA z/OS resource name. The format is either *name/type* or *name/type<</system>>*.

Note that there are a number of special resources that act as *anchors* for dynamic objects that are created by INGPOST. The format of an anchor *resource_name* is ANCHOR/MJR/NameOfAnchor. You define the anchors in the automation policy with the NMC DEFINITIONS policy item. For more details, see *IBM Tivoli System Automation for z/OS Defining Automation Policy*.

MINOR

Specifies the minor resource name associated with the notification

minor_name

This is, more or less, a free-form field for specifying the minor resource name. The value must be valid as part of a RODM object name and may have requirements placed upon it by your BLDVIEWS implementation.

If not specified, the minor resource name defaults to null.

EVENT

Specifies a short 'status' value that is appended to the resource's DisplayResourceName.

event_text

Is a single word that will be folded to upper case and appended to the DisplayResourceName for the corresponding RODM object. It should be informative and short.

If not specified the event text will be null and only the resource name and the minor resource name will be present in the DisplayResourceName.

COLOR

Specifies the new DisplayStatus to be posted for the object within RODM.

GREEN or 129 or OK

Sets the object's DisplayStatus to Satisfactory

LIME or 144

Sets the object's DisplayStatus to Medium Satisfactory

CYAN or 145

Sets the object's DisplayStatus to Low Satisfactory

WHITE or 131 or TRANSITION

Sets the object's display status to Intermediate

YELLOW or 161 or WARNING

Sets the object's DisplayStatus to Low Unsatisfactory

PINK or 160

Sets the object's DisplayStatus to Medium Unsatisfactory

RED or 130 or ALERT

Sets the object's DisplayStatus to Unsatisfactory

UNKNOWN or 132

Sets the object's DisplayStatus to Unknown

Note that the syntax diagram does not show the numeric values for the statuses. However, they are accepted and processed correctly.

There are some special values that trigger different processing.

DELETE

Can only be used against a minor resource. It will delete a single minor resource object that is associated with the major resource.

DELETE_ALL

Should be used with a major resource and deletes all minor resource objects associated with that major resource.

SUSPEND and UNSUSPEND

Changes the setting of the 'suspended from aggregation' part of the object.

AUTO and UNAUTO

Changes the setting of the 'automation in progress' part of the object.

HIDE and UNHIDE

Changes the setting of the 'excluded from exception views' part of the object.

The second part of the value is a set of up to 4 SCL flags. The first set applies to the 'operator marked' part, the second to the 'automation in progress' part, the third to the 'suspended from aggregation' part and the fourth to the 'exclude from exception views' part. The following are valid flags:

- S** Sets the bit
- C** Clears the bit
- L** Leaves the bit unchanged

These flags reduce the number of updates needed to perform a status change. Typically, when a resource changes from, for example, Awaiting Automation to Automation In Progress, you need to:

1. Change its color and status
2. Clear its 'operator marked' bit
3. Set its 'automation in process' bit

With these bits, the update can be done with one single call, posting the status as 144/CS. Without these flags this would require three separate calls. When the resource status changes from Automation In Progress to Satisfactory or Degraded, one would post statuses of GREEN/LC or WHITE/LC.

MESSAGE

Specifies a message to be shipped with the status update. It ends up in the DisplayResourceOtherData field.

message_text

This is the text of the message.

The maximum text length is 140 characters.

If it is not specified, and if a message is available from the safe, that message will be used.

SAFE

This specifies the name of the safe that holds the message that the default message text is to be taken from.

safe_name

This is the name of the safe.

If not specified the default safe (called *) will be used.

If the safe is empty, the default message text is null.

D3 When specified this populates the Data3 field on NMC with the given string.

string

This is the string that populates the Data3 field.

TAG

This can be specified when posting a minor resource against a major resource.

tag

This is the tag that is attached to the object in RODM (as an index). The same tag value can subsequently be used to restrict the scope of a STATUS=DELETE_ALL call for the major resource, so that it will only delete all attached minor resources with the same major resource.

INGQRY

Purpose

The INGQRY REXX function returns the value of the specified attribute for a particular resource.

Syntax

```
Value=INGQRY(res_name,attribute [, 'VERBOSE=NO' ] [, 'VERBOSE=YES' ] )
```

Parameters

res_name

The name of the resource. This value can be a subsystem name or job name. The search sequence is:

1. Subsystem name
2. Job name

attribute

The name of the attribute. It is one of the following (note that, for different resources, not all of these may be available):

APPL

Returns the subsystem name of the resource. This assumes that the specified resource name is a job name.

ASID

Returns the address space ID of the resource.

CATEGORY

Returns the resource category, such as JES2,IMS,DB2.

CMDPFX

Returns the command prefix of the resource.

FILE

Returns the file information that represents the resource.

FILTER

Returns information about the command parameters that are specified to make the process unique.

IPSTACK

Returns the IP stack name of the resource.

|
|

INGQRY

JOB

Returns the job name of the resource.

JOBTYPE

Returns the job type of the resource (MVS, NONMVS or TRANSIENT).

OPER

Returns the work operator that is associated with the resource.

OWNER

Returns the owner information of the resource.

PARENT

Returns the parent information for the resource. Parent information is derived from the HasParent relationship that has a sequence number assigned to it.

PATH

Returns the information about what UNIX process the resource represents.

PID

Returns the USS Process ID (PID) that is associated with the resource.

PLEX

Returns the name of the plex that is associated with the resource. Currently, this attribute is only used for the IMSplex name as specified in the IMS CONTROL Policy Item.

PORT

Returns the TCPIP port that is represented by the resource.

PROCESS

Contains START or STOP if the resource is in the startup or shutdown phase.

STAT

Returns the agent status of the resource.

SUBCAT

Returns the subcategory of the resource, for example, AOR or TOR for CICS.

SUBID

Returns the MVS subsystem identifier of the resource.

SYMBOL_{*n*}

Returns the requested application symbol, where *n* is 1–9.

USER

Returns the USS user ID that is associated with the resource.

WLMNAME

Returns the WLM resource name that is associated with the resource.

WTOR

Returns the outstanding reply IDs.

A null value is returned if an unknown subsystem name, job name or variable name is given, or a syntax error is encountered.

VERBOSE

NO This is the default. The returned value is null.

YES

Causes an error message to be issued if an unknown subsystem name, job name or variable name is given, or a syntax error encountered.

Usage

INGQRY can only be used in the REXX environment.

Examples

The following example obtains the status of resource TSO:

```
stat = INGQRY('TSO','STAT')
```

The following example obtains the subcategory of resource TDBDB001 and a message is issued if *resname* is unknown or a syntax error is encountered.

```
resname = 'TDBDB001'
subcat = INGQRY(resname,'SUBCAT','VERBOSE=YES')
```

INGRCHCK**Purpose**

The INGRCHCK command checks whether a particular resource is in a specific state by checking the observed status of the resource. If the resource is not in the expected state, INGRCHCK waits until the resource reaches this status.

Syntax

```
►►—INGRCHCK—resource—OBSERVED=status—┌──────────────────┐
└──────────────────┘
                └─ INTERVAL=nnn ─┘
```

Parameters*resource*

This is the name of the resource. It can be either a subsystem name or a resource name in automation manager notation.

OBSERVED

This is the observed status that the resource should be in. You can specify more than one status. If you specify more than one status, they must be separated by a blank character and enclosed in parentheses. The observed status can be abbreviated, for example, SO for softdown.

INTERVAL

This specifies the time interval that the routine uses to periodically check whether the resource is in the expected state. The default is 5 seconds. The maximum is 999 seconds.

Examples

```
INGRCHCK AM2/APL/AOC8 OBSERVED=(SO HA)
```

INGRCLUP

Purpose

The INGRCLUP command is used to cancel address spaces that may be left over by a resource that did not properly shut down. Multiple address spaces with the same name can be cancelled.

Syntax

```

|
|  >>—INGRCLUP—jobname—┌ type ─┐ ┌ command ─┐
|

```

Parameters

jobname

The job name of one or more address spaces that must be canceled. The job name can contain wildcard characters except for the first character. An asterisk (*) matches a string of arbitrary length and a percentage sign (%) matches a single character.

type

An optional parameter that indicates the type of address space. It can be one of the following:

- A** ATX
- J** Job
- M** Mount
- S** Started task
- *** System address space

command

This is the command to be issued. The default is Cancel.

Restrictions and Limitations

Primarily INGRCLUP is meant to be called from within the automation policy (that is, the SHUTDOWN policies).

The percentage sign (%) cannot be used as a wildcard in the first character of *jobname*.

Return Codes

- 0** Processing was successful.
- 4** Parameters are invalid.

Examples

The following example cancels all REXX system address spaces:

```
INGRCLUP AXR0* *
```

INGRDS

Purpose

The INGRDS command provides Relational Data Services (RDS) exclusively for the System Automation environment.

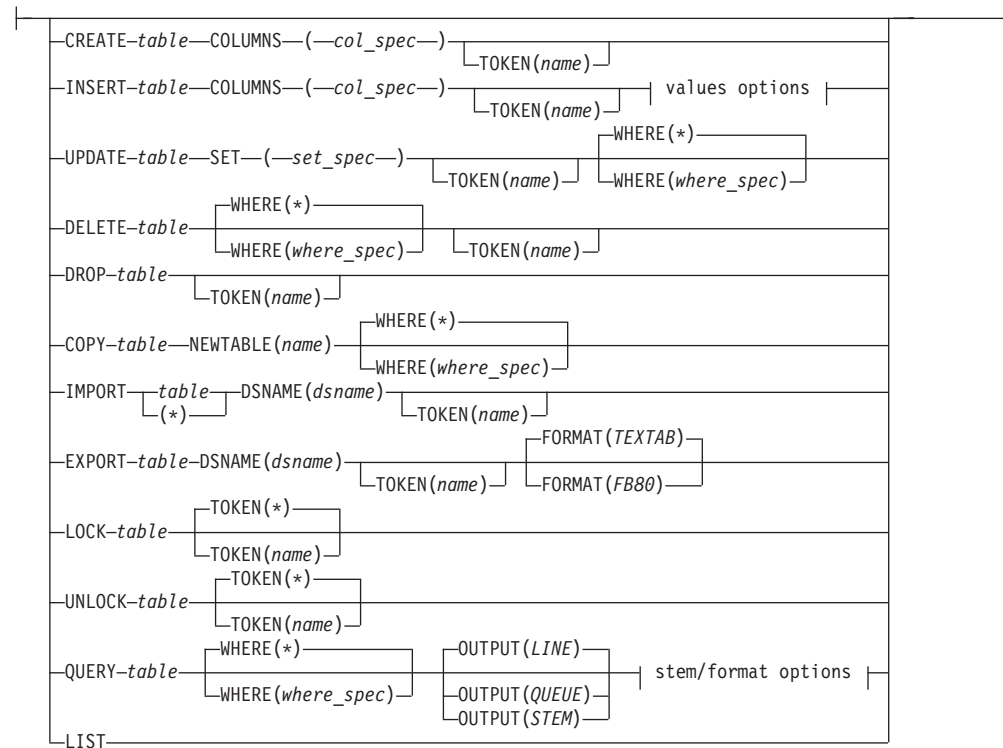
System Automation now offers basic access methods for relational data services (RDS). It is close to the concept of SQL but without the full SQL language parser and with a built-in for System Automation store of its own. A database such as DB2 cannot be attached.

The supporting System Automation command is INGRDS. The entity maintained by INGRDS is a relational data service table, short RDS table or relational data table. The relational data service tables are held in GETMAIN storage of the NetView address space. The available amount of free memory in the NetView address space limits the amount of data that can be stored by RDS. Persistency is achieved by a separate task (see the chapter "Enabling Relational Data Services (RDS)" in *IBM Tivoli System Automation for z/OS Customizing and Programming*). For example, the tables will be periodically saved every 30 seconds into a VSAM KSDS file (see DD INGEMUGL). The tables will be restored during System Automation initialization when NetView and System Automation is started.

Syntax

►► INGRDS | function options |

function options:



stem/format options:



values options:

|—VALUES—(—*val_spec*—)

Parameters**Function/Command**

Available functions are:

CREATE

Creates a relational data table and defines the column definition. Same characters as for OPSVALUE.

INSERT

Inserts a row into the table. If a column is not specified it will be NULL.

DELETE

Delete row(s) into the table. The WHERE clause is a filter for rows to be deleted.

UPDATE

Update existing row(s). The WHERE clause is a filter and SET overwrites the columns.

QUERY

Displays a table with specified columns. Default is columns(*).

DROP Deletes the entire table and all data of this table.

COPY Copies an existing table into a non-existent new table.

IMPORT

Imports a table from a backup data format.

EXPORT

Exports a table into a backup data format.

LIST Shows a list of existing tables.

LOCK Locks a table in order to protect it against changes.

UNLOCK

Removes the lock.

Table

The table name is a string of up to 30 characters. The same character set as defined for OPSVALUE.

col_spec

For function CREATE the syntax is : COLUMNS (Colname1 Char(n1), Colname2 Char(n2),...) where colname is an alphanumeric string with some special characters but without blanks. n1, n2 .. is a number that defines the maximum length of the characters. Only data type Char is supported. For function INSERT and QUERY syntax is : COLUMNS (Colname1, Colname2,...)

val_spec

For function INSERT syntax is: VALUES ('string1','string2',.....). The sequence of string correlates to the sequence of columns specified in col_spec. Omitted values are replaced by NULL if denoted by two commas.

set_spec

For function UPDATE syntax is: SET (Colname1='string1',
Colname2='string2',...)

WHERE

For function QUERY, INSERT, COPY, and UPDATE, see “Syntax of WHERE clause” on page 132.

OUTPUT

Defines how the output of the function QUERY will be returned.

QUEUE

Writes the output lines into the external data queue.

STEM Writes the output lines into stem specified with parameter STEM('out.'). Default stem name is 'rdsout'.

LINE Writes the output lines to CONS ONLY. This is useful for PIPE. LINE is default.

FORMAT

Defines the data format of the output lines.

TEXT Writes the query output in text format padded with blanks until maximum column length is reached.

TEXTHDR

Like TEXT, but with two additional headlines.

TEXTAB

Like TEXTHDR, but with one additional descriptor.

FB80 Fixed Block 80 format (FB80) should be specified only for INGRDS EXPORT and not INGRDS QUERY.

STEM

Defines the name of the stem when needed in conjunction with function QUERY.

DSNAME

Defines the name of a data set when needed in conjunction with function IMPORT/EXPORT.

NEWTABLE

Defines the name of the new table used by function COPY.

TOKEN

Defines the Lock token.

Examples

Using the command INGRDS

```
INGRDS CREATE mytable COLUMNS( NAME Char(30),FIRST_NAME Char(20),CITY Char(10))
INGRDS INSERT mytable COLUMNS( NAME, FIRST_NAME, CITY )
                        VALUES('Smith', 'John', 'New York' )
INGRDS QUERY mytable COLUMNS( NAME, FIRST_NAME, CITY ) WHERE(*)
INGRDS DROP mytable
INGRDS DELETE mytable WHERE NAME='Smith')
INGRDS UPDATE mytable WHERE(NAME='Smith') SET(CITY='Los Angeles')
```

Restrictions and Limitations

The implementation of RDS has some restrictions:

- the number of possible tables is limited to around about 250

INGRDS

- only the data type CHAR is supported (which is stored as variable string internally)
- maximum length of a table row is restricted to 32000 bytes
- less complex WHERE-clause(s) are supported (see "Syntax of WHERE clause" on page 132)
- primary key is not supported. Therefore a table row could be inserted twice. No checking for unique key
- keywords such as WHERE, SET, for example, must follow a bracket. Blanks between keyword and bracket are not recognized
- rows are not sorted since INGRDS does not support a primary key. Therefore function DELETE, UPDATE and QUERY processes all matching rows.

Usage

Hints and Tips

1. Within VALUES() and SET() you may set a hex string such as '00'X or '01020A0B'X.
2. Within WHERE() you may search for a hex string such as '00'X or '01020A0B'X.
3. The command INGRDS makes a difference between the empty string and NULL.
A table cell is empty when the value has length zero.
A table cell is NULL when the value is one byte hex zero, REXX '00'X.
An un-initialized table cell has value NULL.

CREATE a Table

This example creates a table with name TAB1 (the table name is translated to uppercase). A pair for column name and char(nn) is required separated by a comma. The data of a column is always mixed case and can also be hex data. Maximum data length is 10 bytes in this case for all columns.

```
INGRDS CREATE tab1 COLUMNS (col1 char(10), col3 char (10))
```

INSERT Row into a Table

Examples 1 and 2 insert a new row with col1='c11', col2='00'X and col3='c13'.

Example 3 inserts a new row with col1='c11', col2='c12' and col3='c13'

Example 4 inserts a new row with col1='c11', col2='00'X and col3='c13'

1. INGRDS INSERT tab1 COLUMNS(col1 col3) VALUES('c11','c13')
2. INGRDS INSERT tab1 COLUMNS(col1 col2 col3) VALUES('c11','c13')
3. INGRDS INSERT tab1 COLUMNS(col1 col2 col3) VALUES('c11','c12','c13')
4. INGRDS INSERT tab1 COLUMNS(col1 col2 col3) VALUES('c11','00'X,'c13')

DELETE Row from a Table

Example 1 deletes **all rows** where col3 has the specified value.

Example 2 deletes rows where col3 is the empty string.

Example 3 deletes rows where col2 is the NULL string.

1. INGRDS DELETE tab1 WHERE(col3='c13')

- ```
|
| 2. INGRDS DELETE tab1 WHERE(col3=' ')
| 3. INGRDS DELETE tab1 WHERE(col2='00'X)
```

### UPDATE Row of a Table

The examples update all rows where col1 has the specified value.

Example 1 updates col2 to c22 at all rows where col1 is the c21.

Example 2 updates col2 to '00'X (NULL) at all rows where col1 is the c21.

- ```
|
| 1. INGRDS UPDATE tab1 SET(col2='c22') WHERE(col1='c21')
```
- ```
|
| 2. INGRDS UPDATE tab1 SET(col2='00'X) WHERE(col1='c21')
```

### DROP a Table

Dropping a table removes all data from the GETMAIN storage immediately. Since normally the archive command is scheduled periodically every 30 seconds (see resource RDSARCH in *IBM Tivoli System Automation for z/OS Customizing and Programming*) the data that is related to the dropped table will also be removed from the VSAM file at this time.

**Note:** If you recycle NetView after dropping a table and before the archive command, running the DROP command may have no permanent effect and the table might be restored again during SA initialization.

A table might have a lock. In this case you cannot drop the table (example 1) and you must specify the token associated to the lock. Use command "INGRDS LIST" to find the tables and the associated name of the locks. Example 2 drops table TAB1 with lock NET\$BDOW.

- ```
|
| 1. INGRDS DROP tab1
```
- ```
|
| 2. INGRDS DROP tab1 TOKEN(NET$BDOW)
```

### COPY a Table

You can copy an existing table to a new table with a different name. The new table must not exist. If the table has a lock, the lock will be copied as well.

- ```
|
| 1. INGRDS COPY tab1 NEWTABLE(tab2)
```

LOCK and UNLOCK a Table

You can create a table and assign it a lock, see example 1.

The lock protects the table from modification. It does not protect the table being unlocked by another user if the user knows the token. Note, the default token is JOB\$userid (TSO) or NET\$userid (NetView®).

Example 1 shows how to assign a lock to an existing table.

Examples 2 and 3 create a table with lock and removes the lock from it.

- ```
|
| 1. INGRDS LOCK tab1 TOKEN(mylock1)
```
- ```
|
| 2. INGRDS CREATE tab2 TOKEN(mylock2) COLUMNS(...)
```
- ```
|
| 3. INGRDS UNLOCK tab2 TOKEN(mylock2)
```

### Invoking from TSO

You can run all functions from command INGRDS on TSO.

## INGRDS

For that purpose, the command INGRCRDX serves as the client API to the remote NetView server. It sends the corresponding INGRDS command to NetView and receives the response. Output is written to the TSO terminal:

General Syntax: INGRCRDX function table list\_of\_keywords

For example:

```
INGRCRDX QUERY tab1
```

```
INGRCRDX CREATE tab1 COLUMNS(col1 char(1), col2 char(20))
```

### Syntax of WHERE clause

The command INGRDS has some restrictions with the WHERE clause in comparison to full SQL.

#### Restrictions

- Join, subquery and aggregate functions are not supported.
- Host variables are not supported.
- Only a limited set of functions are supported.
- REXX rules apply for expressions.

#### REXX Where-Clause Expression

The finally parsed and resolved WHERE-clause, `resolved_where_clause`, is treated as a string that will be executed by the REXX interpreter via the following REXX instruction:

```
if (resolved_where_clause) then
 bool=1
else
 bool=0
```

#### The REXX rules for expressions and operators apply

For example the operator AND has higher priority than OR. Also the rules for how parentheses are handled is defined by the REXX rules. Restrictions that apply due to the usage of this technique can never be resolved!

#### Restricted Syntax Checking

The syntax parser finds that a column name is invalid/not known, for example, `"xxx='abc'"`:

```
ING338I Function or command INGRCRDS failed, RC=116 REASON=QUERY
TABLE MYTAB Invalid column name xxx
```

In many cases the syntax parser detects that a quote is missing, for example, `'xxx='abc '"`:

```
ING338I Function or command INGRCRDS failed, RC=115 REASON=QUERY
TABLE MYTAB WHERE/SET=clause parser error 46
```

But for many other invalid WHERE-clauses the SA syntax parser is less restrictive.



For example, if the operator is not valid or not specified at all such as "name 'abc'" (rather than " name='abc'") or "name like %abc%" (rather than "name like '%abc%'") the final expression results in a REXX error.

If such a REXX error occurs in `resolved_where_clause` then the message ING338I reports the error.

#### Error Message examples:

- "'1'/2" results in:  
 ING338I Function or command INGRCRDS failed, RC=116 REASON=QUERY TABLE MYTAB  
 No match due to bad WHERE-clause: Logical value not 0 or 1
- "name like %aa%" results in:  
 ING338I Function or command INGRCRDS failed, RC=116 REASON=QUERY TABLE MYTAB  
 No match due to bad WHERE-clause: Invalid expression
- "name like abc" results in:  
 ING338I Function or command INGRCRDS failed, RC=116 REASON=QUERY TABLE MYTAB  
 No match due to bad WHERE-clause, error near: abc

#### Supported Syntax

►—WHERE( [NOT]{predicate} [AND|OR [NOT] predicate] )—►

#### Examples:

```
WHERE(NAME='Bond' AND CITY='London')
```

The following list of expressions can be used to construct the WHERE clause:

1. **Predicate** is an expression that can be true or false and can contain either uppercase or lower case characters. You can specify the following types of predicates:
  - Comparison predicates
  - IN predicate
  - LIKE predicate
2. **Character function** such as substring function. The substring function allows you to retrieve parts of character strings that are to be used as search criteria. The substring function and the lower/upper function may be used in conjunction with the predicates previously mentioned.
3. You can combine predicates and the character functions into compound **Boolean Expressions**. For example, you can create a WHERE clause like the following:
 

```
(NAME = 'BOND' and CITY <>BIRTHCITY) OR FIRST_NAME = 'James'
AND STREET IN('Central Avenue','Lincoln Road','Catwalk')
```

#### Comparison Predicate

A comparison predicate uses the syntax: `colname relationaloperator value`

The **relationaloperator** can be any of the following:

Table 7. Relational Operator Listing

| Relational Operator | Description              |
|---------------------|--------------------------|
| =                   | equal                    |
| <>                  | not equal                |
| >                   | greater than             |
| >=                  | greater than or equal to |
| <                   | less than                |
| <=                  | less than or equal to    |

Example: DESIRED\_STATE <> 'down'

### IN Predicate

An IN predicate compares the column name to one or more strings. The maximum number of strings is 20. Use the following syntax to specify an IN predicate:  
colname IN (string1,string2,...,string20)

For example, NAME IN('Ashton','Bond','Connory','Jones','Tong')

### LIKE Predicate

A LIKE predicate compares a column to a wildcard value you specify and selects columns conforming to the wildcard value. Use this syntax for a LIKE predicate:  
colname LIKE 'wildcard'

You must enclose the wildcard value in single quotes. A wildcard can include any combination of characters and either the percentage sign (%) or the underscore(\_).

The percentage sign denotes any set of characters, including blanks.

For example, the following clause: NAME LIKE '%on%' selects from the NAME column all of the following values: Ashton,Bond,Connory,Jones,Tong

The **underscore sign** denotes only one character in a specific position. For each character of column data you do not want to match, the wildcard must include an underscore. For example, the following clause: NAME LIKE '\_on\_' selects from the NAME column that contains all of the following values: Bond, Tong

### Restrictions for ESCAPE keyword

- ESCAPE character cannot be specified. It is always backslash (\)
- The ESCAPE keyword can be specified but will be ignored.

When the percentage sign (%) or underscore character is part of the column data, use the ESCAPE character backslash \. The ESCAPE character prevents INGRDS from interpreting the percentage sign (%) and underscore characters in column text as wildcards. For example, to find the columns containing the characters I\_MS1 and I\_MS2, use the following predicate: NAME LIKE 'I\\_MS\_'

### Character Functions

The operands of a character string function can be literal character strings or column names. The following character functions are supported:

- SUBSTR (*argument* FROM n FOR m)
- UPPER (*argument*)
- LOWER (*argument*)

The *argument* may be either a column name or a literal.

Examples:

```
SUBSTR (NAME FROM 3 FOR 2) = 'nd'
```

```
UPPER (NAME)= UPPER('bond')
```

The **SUBSTR** clause takes from the column name argument starting at position n as many characters as specified with n. For example SUBSTR (CITY FROM 5 FOR 4) returns the string 'York' if the CITY is 'New York'.

The **LOWER** function translates a character string literal or column value to all lower case characters.

The **UPPER** function translates a character string literal or column value to all upper case characters.

### Example of a Complex WHERE-clause

The purpose of this WHERE-clause is simply to show how to combine predicates. It shows an example of combinations of comparison predicates, IN and LIKE predicates and character functions SUBSTR and UPPER.

```
table = 'sample'

/*define query*/
where_clause =,
"(NAME='Bond' AND CITY <> 'London')",
"OR ('James' <> FIRST_NAME)",
"OR NOT (CITY='Berlin')",
"AND CITY IN ('Paris','London','New York')",
"AND FIRST_NAME LIKE '%ame%'",
"AND SUBSTR (CITY FROM 4 FOR 3)='don'",
"AND UPPER (NAME)=UPPER('james')",
"AND LOWER (NAME)='james'"

/*execute query*/
'INGRDS QUERY' table 'WHERE('where_clause ') OUTPUT(STEM) STEM(out.) FORMAT(TEXT)'

/*show query output*/
if(rc=0) then do
 do i=1 to out.0
 say out.i
 end
```

### The RDS Table Output

A RDS table can be retrieved via the subcommand QUERY. The parameter OUTPUT defines the output destination and the parameter FORMAT defines the data format. For example, a table can be displayed on the NetView console via command:

```
INGRDS QUERY tab1 WHERE(*) OUTPUT(LINE) FORMAT(TEXTHDR)
```

**TEXT FORMAT:** Here you see an example of FORMAT(TEXTHDR). The first two lines are the headline followed by the table rows.

## INGRDS

The first line defines the column names as specified in the subcommand CREATE.

The second line defines the maximum length of a column. However if the column name is longer than the maximum length, then the length of the column name is used.

The table row lines display the cell data padded with blanks up to the maximum column length.

| NAME    | FIRST_NAME | CITY      |
|---------|------------|-----------|
| Bond    | James      | London    |
| Smith   | John       | New York  |
| Schulz  | Emma       | Berlin    |
| Metzger | NULL       | München   |
| Wagner  |            | Frankfurt |

Note that INGRDS makes a difference between "empty string" and "NULL".

An empty string is a character string with length 0, for example the FIRST\_NAME of Wagner is the empty string. But the FIRST\_NAME of Metzger is NULL.

**Viewing a RDS Table within NetView:** You may want to use the NetView window command in order to display a RDS table using the following command:

```
WINDOW INGRDS QUERY MYTAB FORMAT(TEXTHDR)
```

### List of Existing Tables

You can get a listing of all existing tables via command:

```
INGRDS LIST
```

- 1st column of the listing is the table name
- 2nd column of the listing is the internally used pool name
- 3rd and 4th column of the listing is the number of rows and columns of the table
- The last column is the list of column name.

The command output looks like following (list of column names is truncated):

| TABLE NAME | POOL     | ROW | COL | LOCK      | COLUMNS        |
|------------|----------|-----|-----|-----------|----------------|
| TA1        | !RD!TA1  | 124 | 015 | JOB\$BDOW | RES JOB OWN TY |
| TA2        | !RD!TA2  | 005 | 015 | UNLOCKED  | RES JOB OWN TY |
| JOBINFO    | !RD!JI08 | 005 | 007 | UNLOCKED  | CICSID VS DS C |
| MYAUDIT    | !RD!MUT1 | 073 | 023 | UNLOCKED  | DSN CDT CNT VO |
| SYSTASKS   | !RD!SAS8 | 127 | 011 | UNLOCKED  | JOBNAME TASK_S |
| WINDOWS    | !RD!WDS7 | 162 | 007 | UNLOCKED  | JOBNAME STATUS |
| X1         | !RD!X1   | 000 | 002 | NET\$BDOW | COL1 COL2      |

### Obtain and Release Table Lock

The command INGRDS allows you to obtain and release a lock for a specific table. A lock is a token with a maximum length of 16 bytes. A lock token must not contain blanks. A lock token cannot be the empty string.

The token 'UNLOCKED' is reserved and should not be used as lock token.

The purpose of the table lock is to protect a table from modification while another user has access to it. For example if you edit the table on TSO via command 'INGRCRDY EDIT table' the table will be locked with the corresponding TSO user id.

If a table already has a lock the following subcommands are rejected with return code 102 if the lock does not match the token specified via parameter TOKEN:  
DROP, INSERT, UPDATE, DELETE, IMPORT

For example, the following command deletes the table if the table has the lock 'JOB\$ABC':

```
INGRDS DROP table TOKEN(JOB$ABC)
```

**Note:** The lock token associated with a table is much less restrictive than, for example, an ENQ lock. The lock token is not associated with a task or an address space. It is like an attribute of the table that can be set or removed. Any NetView operator or TSO user (that is allowed to use INGRCDY) can set or remove the lock token. Therefore it might be possible (but not recommended) by a NetView operator or a TSO user userid to set manually the lock token to JOB\$userid2, where userid2 is a foreign TSO userid. Of course, in this case, the TSO user userid2 is able to access and overwrite the table although it was locked by userid1. Also it is recommended that you do not use a lock token that starts with NETV\$ or JOB\$ yourself.

**Obtaining a Lock:** You can obtain a lock for an existing table via the following commands:

```
INGRDS LOCK table [TOKEN(*|token)]
INGRDS CREATE table [TOKEN(*|token)]...
INGRDS EXPORT table [TOKEN(*|token)]...
INGRDS IMPORT table [TOKEN(*|token)]...
```

**Release a Lock:** You can remove the table lock via following command:

```
INGRDS UNLOCK table TOKEN(*|token)
```

When importing a table from a data set it is also possible to remove the lock via command:

```
INGRDS IMPORT table TOKEN(UNLOCKED)...
```

**Default Lock:** The default lock token is:

- for NetView NETV\$xxxx
- for TSO JOB\$xxxx

where xxxx is the NetView task id or TSO user id.

If you omit the parameter TOKEN for subcommand LOCK or UNLOCK the default is assumed.

If the parameter TOKEN is not specified for any other subcommand then no default token is assumed. In this case you must specify parameter TOKEN if needed.

If you specify the parameter TOKEN with value \* then the default token is used.

### Import/Export SQL Tables

With the command INGRDS you may backup a specific RDS table and restore it later on.

## INGRDS

One of the following data formats is supported:

1. FB80 format - may be used to backup an RDS table into a data set with fixed block 80 format.
2. TEXTAB format - may be used to import easily external data into an RDS table. Data layout is the same as used for 'INGRDS QUERY table FORMAT(TEXTAB)'. Refer also to 'SA TSO Table Editor INGRCRDX' section in the "Enabling Relational Data Services (RDS)" chapter of *IBM Tivoli System Automation for z/OS Customizing and Programming*.

**Note:** The import/export data set must exist already before invoking the command INGRDS.

**Import Function:** The command syntax that imports a SQL table is as follows:

```
INGRDS IMPORT table | * DSNAME(dataset)
```

The data set name might be in quotes but also no quotes are accepted. It must be a fully qualified data set name.

The command INGRDS reads the data format identifier from the first record of the data set. If the format checking is successful then INGRDS reads the data from the data set and extracts the table name, the column names and the size of the columns. It creates the table and inserts each row. If the table already exists it will be deleted first.

If the table name is specified then the import data is used to create a table with the name specified in the INGRDS command. If the table name \* is specified then the table name from the import data is used.

**Export Function:** System Automation exports a table into a data set with the same data layout as required by the import function.

The command syntax that exports a SQL table is as follows:

```
INGRDS EXPORT table DSNAME(dataset) FORMAT(FB80|TEXTAB)
```

The logical length of the data set or member of the PDS must be large enough to hold the maximum length of an output record. This is 80 byte for FORMAT(FB80) but could be much larger for FORMAT(TEXTAB).

---

## INGRTCMD

### Purpose

The INGRTCMD command can be used as a second level NMC command exit for issuing commands from NMC. It takes an object ID and a command string and substitutes object parameters into the command string before routing it to the appropriate system for execution.

### Syntax

```
▶▶—INGRTCMD—object_id—cmd_string—▶▶
```

## Parameters

### *object\_id*

Is the RODM object ID that the command should be issued against. It is used to determine the substitution parameters as well as the target sysplex for the command. The command is sent to the system within the target sysplex that the currently received heartbeats and status change notifications originate from.

### *cmd\_string*

Is the command to be issued. It may include substitution tokens.

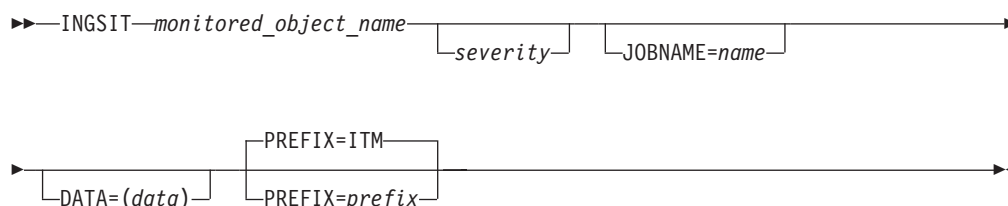
## INGSIT

### Purpose

The INGSIT command allows you to report an event to SA z/OS. To allow SA z/OS to react to that event, it must be associated with one or more Monitor Resources through the monitored object name and optionally a job name. Refer to *IBM Tivoli System Automation for z/OS Customizing and Programming* for a description of Monitor Resources and the concept of a monitored object.

The event information consists of the monitored object name, an optional event severity, an optional job name, and optional related data. INGSIT transforms the event into a well-formed ING150I message that can subsequently be automated on behalf of the monitored object according to the definitions in the automation policy.

### Syntax



## Parameters

### *monitored\_object\_name*

This is the name of the monitored object without the prefix associated with one or more Monitor Resources. To report IBM Tivoli Monitoring situations to SA z/OS (for example, from OMEGAMON XE for z/OS) use the situation as the monitored object and specify its name *in uppercase* with a prefix of ITM in the MONITOR INFO policy when defining the Monitor Resource.

The monitored object name, including the prefix, can be up to 50 characters long and consist of any character except any of the following characters: ' " , ( ) = ? \* and blanks. The monitored object name is case-sensitive.

### *severity*

This is the severity associated with the event. If it is omitted, the text string WARNING is used as the default severity. To report IBM Tivoli Monitoring situations to SA z/OS (for example, from OMEGAMON XE for z/OS) you can use the alert severity that was optionally assigned to the situation or any other severity string if the default does not meet your requirements.

When processing message ING150I via the NetView Automation Table, the severity string is passed as the CODE1 parameter to the INGMON generic monitoring command. If you want to map this severity to a health status, you have to ensure that there is a matching CODE entry in the MESSAGES/USER DATA policy for message ING150I. The health status, if any, is specified as the second word in the value returned for that CODE.

If there are several different severities for the same monitored object and you want to react in different ways to such an event, you also have to ensure that there is a matching CODE entry in the MESSAGES/USER DATA policy for message ING150I. The first word in the value returned is then treated as the command selection. Alternatively, return a # to perform PASS processing for ING150I, or #*usermsg* to perform independent PASS processing for the message entry, *usermsg*.

Refer to *IBM Tivoli System Automation for z/OS Customizing and Programming* for examples of how to set up your automation policy to react to message ING150I as described.

**JOBNAME=*name***

This is an optional job name. If it is specified, it is compared with the job name specified for a monitored object to match this situation to a particular monitor resource. If a job name is not passed, the event reported through this invocation of INGSIT may only match monitor resources that are defined without a job name.

**DATA=(*data*)**

This is an optional string of additional information that is related to the event. The parentheses are required if you want to pass a string that contains blanks or commas. For a single word, the parentheses can be omitted. *data* is appended to message ING150I without further analysis to be used for any optional user-specified purposes by the automation.

**PREFIX=*prefix***

This is the prefix followed by a period that precedes the specified *monitored\_object\_name*. The default prefix is ITM. The monitored object name appear in message ING150I as: *prefix.monitored\_object\_name*

## Return Codes

- 0 Normal completion.
- 1 Syntax error. Invalid parameters were passed to INGSIT. Refer to the netlog for additional error information.
- 2 The command environment for INGSIT was not appropriately initialized at the time that this command was issued. Possible reasons are that the agent is currently being initialized or a cold start of the automation control file is being performed.

## Restrictions and Limitations

For situations, monitored object names *cannot* be distinguished by case. The situation *must* be specified as monitored object in uppercase in the MONITOR INFO policy.

## Usage

The ING150I message that is generated as a result of the event transformation may or may not cause an automated response. If a monitored object was specified but a Monitor Resource was not found for it, no automation is triggered.



The health status of a Monitor Resource that matches the monitored object name is not changed unless you map the severity (either your own input or the default of WARNING) explicitly to a health status using a CODE entry in the MESSAGES/USER DATA policy for message ING150I.

## Examples

### Example 1: Using the Default Severity

The following example reports the situation OS390\_Local\_PageDS\_PctFull\_Warn to SA z/OS. A default severity of WARNING is used and the related data passed to SA z/OS is substituted from the page data set utilization attribute, which here is 25%.

```
INGSIT OS390_LOCAL_PAGEDS_PCTFULL_WARN DATA=(25)
```

This yields:

```
ING150I 09:34:37 08/01/2007 : ITM.OS390_LOCAL_PAGEDS_PCTFULL_WARN
WARNING N/A DATA= 25
```

### Example 2: Using an Explicit Severity

The following example reports the situation OS390\_PageDSNotOperational\_Warn to SA z/OS. The severity, although it indicates a warning-level situation, is reported as minor and the count of non-operational data sets is passed as the related data, which here is 1.

```
INGSIT OS390_PAGEDSNOPERATIONAL_WARN Minor Data=1
```

This yields:

```
ING150I 09:35:42 08/01/2007 : ITM.OS390_PAGEDSNOPERATIONAL_WARN
MINOR N/A DATA= 1
```

---

## INGSTOBS

### Purpose

The INGSTOBS command lets you subscribe as a status observer for one or more resources. Whenever a status change occurs, the automation manager sends you a notification. The following statuses are applicable:

#### Observed

Is the current status of the resource monitored by the automation manager.

#### Desired

Is the status the resource should be in. The automation manager attempts to put the resource into this state. This is also called the 'goal' of the resource.

#### Automation

Is the current status of the resource within the automation process of SA z/OS.

#### Compound

Is the summary of all statuses and a few other indicators.

#### Startability

Indicates whether the resource is ready to be started when a start command is issued.

#### Health

Application-specific performance and health monitoring provides a separate status to inform you about the application's health.

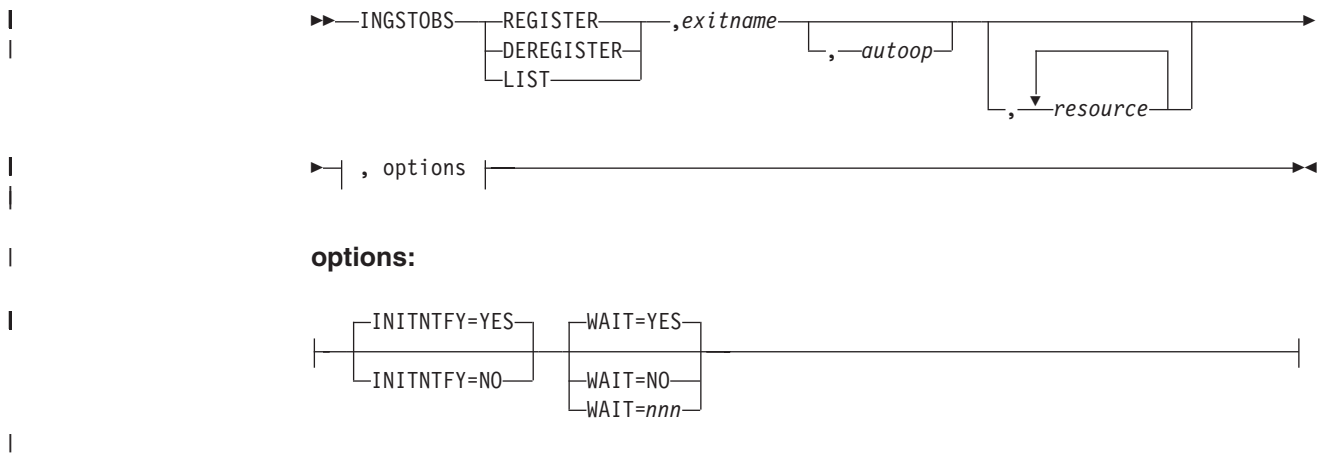
# INGSTOBS

To register as a status observer, specify the name of a REXX exit. SA z/OS invokes this exit for each status change. The following parameters are passed to the exit:

- The name of the resource, for example, TSO/APL/SYS1
- The observed status
- The automation status
- The desired status
- The compound status
- The startability status
- The health status

The parameters are separated by a comma.

## Syntax



## Parameters

### REGISTER

Performs the subscription by linking the specified exit to each resource specified.

Each subsequent status change of the resource triggers the exit to be invoked.

### DEREGISTER

Breaks the link between the exit and the resource. The exit is no longer invoked if a status change of the resource occurs.

### LIST

Displays the resources that are subscribed and linked to the specified exit.

#### *exitname*

Specifies the name of the REXX exit to be invoked when a status change of the resource occurred.

#### *autoop*

The automated function that the autotask name is defined from. The exit is scheduled to run on the autotask associated with the automated function. If omitted, the exit runs on the SA z/OS task responsible for communicating with the automation manager. It is recommended to use a different task.

#### *resource*

Specifies the name of the resource or family of resources, via wildcard, for

example, TSO/APL/\*. The resource names must be separated by a blank. Alternatively, the list of resources can be passed to the command with a NetView default Pipe Safe.

The parameters must be separated by a comma.

#### **INITNTFY**

Specifies whether the automation manager sends the status change immediately:

##### **YES**

The automation manager sends the status change immediately. This causes a call to the user exit immediately.

**NO** This causes the automation manager to send the status change with the next subsequent status change.

#### **WAIT**

Specifies whether to wait until the automation manager has processed the request.

##### **YES**

Wait for completion. This is the default.

**NO** This causes the subscription request to be sent to the automation manager *without* waiting for its completion.

##### *nnn*

This is the number of seconds to wait before giving up and reporting that a timeout has occurred. The maximum time interval is 999 seconds. If omitted, the time interval is 30 seconds.

## **Restrictions and Limitations**

The INGSTOBS command can only be issued for a local system.

## **Examples**

To register TEST2 as a status observer exit for all resources starting with CICS, specify:

```
INGSTOBS REGISTER,TEST2,MSG2OPER,CICS*/APL/AOC8
```

When the exit is invoked it runs on the autotask that is associated with the MSG2OPER automated function.

To display the resources that are associated with the status observer exit TEST2, specify:

```
INGSTOBS LIST,TEST2
```

Figure 7 on page 144 shows exits and associated resources:

| Resource               | Exits...         |
|------------------------|------------------|
| AMSINGLE/APG/AOC1      | INGRYSOS/MSGOPER |
| FEATEMUL/APG/AOC1      | INGRYSOS/MSGOPER |
| MOVSYSTEM/APG/AOC1     | INGRYSOS/MSGOPER |
| PARCHILD/APG/AOC1      | INGRYSOS/MSGOPER |
| PBBB1/APG/AOC1         | INGRYSOS/MSGOPER |
| PBBC1/APG/AOC1         | INGRYSOS/MSGOPER |
| PBMA1/APG/AOC1         | INGRYSOS/MSGOPER |
| SYSSRV1/APG/AOC1       | INGRYSOS/MSGOPER |
| SYSSRV2/APG/AOC1       | INGRYSOS/MSGOPER |
| *** End of Display *** |                  |

Figure 7. Exits and Associated Resources

## INGSTX

### Purpose

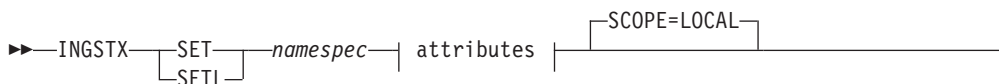
INGSTX controls user-defined status items. Status items can be defined, updated, queried, and deleted. They are stored in the status item repository within the SA z/OS automation manager.

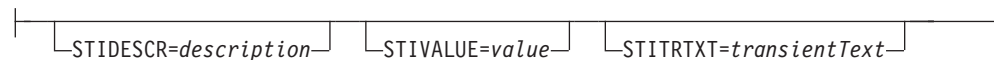
Each status item has the following attributes that are either set or modified through INGSTX:

- STIORIGN — Status item originator. This field contains the name of the system that the status item is collected from. This field cannot be updated.
- STISTGRP — Status item group. If the status item was defined in the form of two subfields separated by a period, this is the first subfield. Otherwise, the field is empty (null string). This field cannot be updated.
- STISTNAM — Status item name. If the status item was defined in the form of two subfields separated by a period, this is the second subfield. Otherwise, this is the complete name that was specified when the status item was created. This field cannot be updated.
- STIVALUE — Status item’s current value. An arbitrary positive or 0 integer value expressing the status item’s current condition. This field can be updated.
- STIDESCR — Status item description. A textual description of the status. This field can be updated.
- STITRTXT — Status item transient user text. This field can be used to store transient user data. This field can be updated.
- STITCHNG — Status item change time. The format of this field is: CYYMMDDHHMMSSmmm. This field is updated by SA z/OS when a status time was set the last time. Format = GMT.
- STIPERST — Status item persistency. This field is set upon first mention of this status item. A value of 1 means it is persistent; 0 means it is non-persistent.

### Syntax

To define or update a status item, use the following syntax:

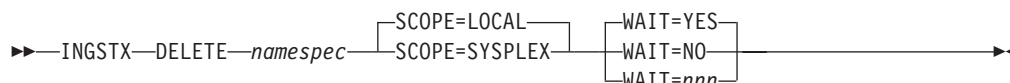


**attributes:**

To query a status item, use the following syntax:



To delete a status item, use the following syntax:



## Parameters

### SET/SETL

This is the function code to define or update a status item. A status item is defined implicitly upon the first mention of the item name. At definition time, you can determine whether the status item should be persistent across SA z/OS sessions or even across IPLs. The PERSISTENT keyword is ignored on subsequent updates of the same status item.

With function code SET the change time in the STITCHNG field is in GMT. If you want to use the local time, use the alternate function code SETL.

### QUERY

This is the function code to query one or more status items that match the pattern denoted by *namespec*. The status item attribute values are positional and are returned in the following format (all on one line):

```
STIORIGNSTISTGRPSTISTNAMSTIVALUESTIDESCR
STITRTXTSTITCHNGSTEPPERST
```

where <del> is a non-ambiguous delimiter character. For an attribute that has not been specified, a null string is returned, that is, <del><del>. STISTGRP is the first substring, if any, of the status item name, and STISTNAM is the remainder, if any, excluding the period.

<del> is the character , which represents X'FF'.

### DELETE

This is the function code to delete one or more status items that match the pattern denoted by *namespec*.

#### *namespec*

This is either the complete specification of a status item or a specification pattern using the asterisk (\*) wildcard character.

Each status item has a system scope. To enable grouping of status items, the name can be divided into two substrings separated by a period (.). For each substring, only alphanumeric characters and the national characters \$, &, and # are allowed. The second substring can contain additional periods for readability purposes. The first character must not be a period or a numeric character. Its maximum length cannot exceed 32 characters.

When a wildcard is used, it can be specified at the beginning or at the end of each substring. The following are examples of valid *namespecs*:

- CICSREGA.MAXTASKS
- CICS\*.MAX\*
- \*.MAX\*
- JOB\*

Status item names are case-sensitive.

**Avoiding the '.' for Status Items with Group names:** If a Take Action command on the Tivoli Enterprise Portal (TEP) refers to a *namespec* of type Group.Name, it is difficult to generate a one word argument with a dot as a separator. You can therefore use a colon (:) as a separator instead, resulting in a *namespec* of type Group:Name. The corresponding notation in the Take Action command is therefore:

```
&Status_Items.Group:&Status_Items.Name
```

#### attributes

Each status item has a set of optional attributes. They are specified in *attribute=value* pairs. If the value is a string that contains blanks, or you want to preserve the case of the characters, enclose value in single or double quotation marks.

Valid attributes are:

#### **STIDESCR**=*description*

A textual description of the status item. The maximum length is 32 characters. Basically, all readable characters are allowed.

#### **STITRTXT**=*transientText*

User-specific text that provides further details about the status item's value. The maximum length is 128 characters. Basically, all readable characters are allowed.

#### **STIVALUE**=*value*

An arbitrary 4-byte positive or 0 (zero) integer value of the status item, up to 2,147,483,647. If not specified, the default is set to 0.

#### SCOPE

This keyword indicates whether the request that is specified addresses status items from the local system only or from all systems in the sysplex (more specifically, all systems connected to the automation manager's XCF group).

#### **SCOPE=LOCAL**

Only status items bound to the local system are addressed. This is the default.

#### **SCOPE=SYSPLEX**

Status items from all systems in the same sysplex are addressed. This scope is not allowed for SET requests.

#### WAIT

This keyword indicates whether the request is executed synchronously or

asynchronously and, if synchronously, how long the caller is willing to wait for an answer. A request will be discarded if the default or user-specified wait time expires.

**Note:** This parameter is always either YES or a time period for QUERY requests.

**WAIT=YES**

The request is executed synchronously, that is, the caller regains control only when the request has been processed by the automation manager, either completely or after waiting for more than 30 seconds. If the request could not be processed within this time, it is discarded. This is the default option.

**WAIT=NO**

The request is executed asynchronously, that is, the caller regains control immediately after the request was accepted by the automation manager. The automation manager executes the request in the background.

**WAIT=nnn**

This option is similar to WAIT=YES with the difference that the maximum time the caller waits for the request to complete is specified as *nnn* seconds. Specify any value between 1 and 999 seconds.

**PERSISTENT**

A status item can be defined as persistent to live across SA z/OS sessions or IPLs. Non-persistent status items are implicitly removed from the status item repository in the SA z/OS automation manager upon disconnecting a system from the sysplex XCF group.

**PERSISTENT=NO**

Status item is non-persistent. This is the default.

**PERSISTENT=YES**

Status item is persistent.

## Return Codes

The following return codes are passed back upon completion of INGSTX:

- 0 Normal completion.
- 3 The operator that invoked INGSTX is not authorized to set, query, or delete a status item.
- 1 Keyword PERSISTENT was specified for an update request of an existing status item. It is ignored.
- 2 Function QUERY or DELETE was specified but no status item was found in the status item repository that matches the given *namespec*.
- 3 The request could not be processed successfully by the automation manager. Refer to the NETLOG for additional error information.
- 4 Syntax error. Invalid parameters were passed to INGSTX. Refer to the NETLOG for additional error information.
- 6 The command environment for INGSTX was not appropriately initialized at the time this command was issued. Possible reasons are that the SA z/OS agent is currently being initialized or a cold start of the automation control file is being done.
- 7 INGSTX failed to create a system resource list for requests with SCOPE=SYSPLEX. Refer to the NETLOG for additional error information.

## Examples

### Example 1

To define two new non-persistent status items called CSA and ECSA, specify:

## INGSTX

```
INGSTX SET CSA stidescr="CSA Below" stivalue=0
 stitrtxt="Utilization 10%"
```

```
INGSTX SET ECSA stidescr="CSA Above" stivalue=0
 stitrtxt="Utilization 12%"
```

### Example 2

To update the status item CSA that was defined in the previous example, specify:

```
INGSTX SET CSA stivalue=10 stitrtxt="Utilization 31%"
```

### Example 3

To query all status items on the local system that end with the three letters CSA, specify:

```
INGSTX QUERY *CSA
```

For the status items defined in examples 1 and 2, this query returns output similar to:

```
SYS1[][]CSA[10]CSA Below[]Utilization 31%[1060203100659000][0]
SYS1[][]ECSA[0]CSA Above[]Utilization 12%[1060203100105000][0]
```

Where [ ] represents X'FF'.

### Example 4

To delete all status items that originated on the local system, specify:

```
INGSTX DELETE *
```

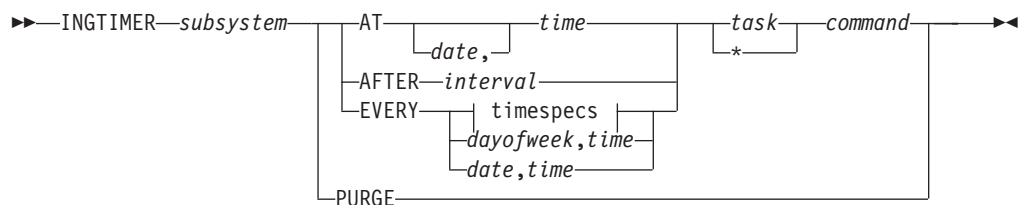
---

## INGTIMER

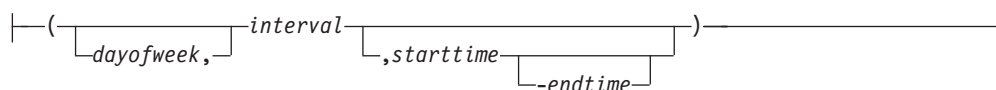
### Purpose

The INGTIMER command links NetView timer commands to subsystems. This means that the timer is only active when the subsystem is active. When the subsystem terminates, the timer commands are automatically purged. To deactivate the timers at SHUTINIT time, you can specify the INGTIMER subsystem PURGE command as a SHUTINIT command.

### Syntax



#### timespecs:





## Parameters

### *subsystem*

Specifies the name of the subsystem.

**AT** Specifies the start time of the command.

### **AFTER**

Specifies the time interval that must elapse after the subsystem became active. When this time interval has elapsed, the command runs. For example, if the subsystem becomes active at 12:00 am and you specify 2 hours, the command runs at 2:00 pm.

### **EVERY**

Specifies the times when the command is to be repeated between the start time and end time.

### **PURGE**

Specifies that all timers associated with the subsystem are purged.

### *date*

Specifies the date, in mm/dd/yy format, that the command should run on. You can specify one or more Xs for the year or both the year and month. The command will then run at the next month or year increment. For example:

```
mm/dd/XX07
mm/dd/XXX8
Xm/dd/XXXX
XX/dd/XXXX
```

### *time*

Specifies the time that the command is to run at. The format is hh[:mm[:ss]]. Instead of entering digits, you can specify one or more Xs at the beginning. If the time begins with an X or multiple Xs instead of a number, the command is set to begin at the next increment of time.

### *interval*

Specifies the time interval that is to elapse before the command runs. The format is hh[:mm[:ss]]. Minutes and seconds are optional values.

### *starttime*

Specifies the start time of the command, which is when it is to be run for the first time. The format is hh[:mm[:ss]]. Minutes and seconds are optional values.

The specified time can be earlier than the current time. The command is then run at the next regular interval after the current time, with intervals calculated based on the start time.

If the time begins with an X or multiple Xs instead of a number, the command is set to begin at the next time increment.

### *endtime*

Specifies the time when the interval is to end. The format is hh[:mm[:ss]]. Minutes and seconds are optional values. Applies only when the interval is shorter than 24 hours.

### *dayofweek*

Specifies the day of the week when the timer command should run. Specify MON through SUN, WEEKDAY, WEEKEND, or ALL.

### *task*

Specifies the user ID of the operator that the command is to be executed from. It can also be an automated function. The default is the work operator that is associated with the subsystem. The timer itself runs on the PPT task.

## INGTIMER

- \* This is a placeholder that indicates that the default is used. The default is the work operator that is associated with the subsystem.

*command*

Specifies the command to be issued when the timer expires.

All timers are converted to the NetView CHRON command format. Thus, daylight-saving-time switching is supported. The timer runs on the PPT task.

**Note:** Storing the timer in the NetView save/restore database is unnecessary because the timer is only active while the subsystem is in an UP state.

## Restrictions and Limitations

None.

## Usage

To link a timer to a subsystem, you must register the NetView timer command as follows:

- At subsystem post-start time (use SHUTDOWN policy of the customization dialog). This associates the timer command with the subsystem and activates the timer.
- Whenever NetView is restarted (use pseudo message ID ACORESTART). This activates the timer command again.
- The timer commands can also be defined under pseudo message ID INGTIMER. In this case they do not need to be specified in the subsystem's POSTSTART and ACORESTART definitions.

The timers are only in effect when the subsystem that they are defined for is active. This is useful for applications that can be moved within the sysplex.

## Examples

To issue a command that should run every 30 minutes between 10:00 am and 2:00 pm, specify the following:

```
INGTIMER TSO EVERY (00:30,10:00-14:00) * F MVS &SUBSJOB,GETLSEQ
```

To issue a command that should run 10 minutes after a certain subsystem became available, specify the following:

```
INGTIMER &SUBSAPPL AFTER 00:10 * MSG,ALL Subsystem is now active
```

To issue a command that should run each Friday at 5:00 pm, specify the following:

```
INGTIMER &SUBSSYS EVERY FRI,17:00 PPT MVS D T
```

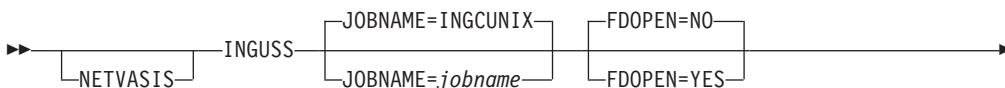
---

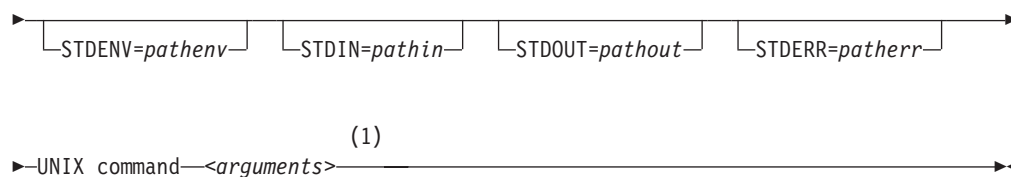
## INGUSS

### Purpose

The INGUSS command allows an automation procedure to send commands to z/OS UNIX System Services.

### Syntax



**Notes:**

- 1 <arguments> may include redirection (see “Examples” on page 153). Redirection arguments are passed to, and processed by, the specified UNIX command and not by INGUSS.

By default the standard streams (fd0, fd1 and fd2) are set to /dev/null. This may cause problems for commands that expect fd0, fd1 and fd2 to be assigned stdin, stdout and stderr (for example, cron). Redirection can be used to bypass this problem (see “Examples” on page 153).

**Parameters****NETVASIS**

Prefix the INGUSS command with NETVASIS if you want to pass the command text or the various path specifications (STDIN,STDOUT,STDERR,STDENV) in lower or mixed case.

**JOBNAME=jobname**

This is the MVS job name used for the newly created address space that runs the specified command. If you do not specify a job name, INGCUNIX is the default.

**FDOPEN**

This parameter is used to determine whether INGUSS opens STDOUT and STDERR before invoking the specified UNIX command.

**NO (DEFAULT)**

STDIN, STDOUT and STDERR are not opened.

**YES**

STDOUT and STDERR are assigned to /dev/console and opened, unless otherwise specified.

**STDENV**

This optional parameter is used to specify a path for STDENV.

**pathenv**

The path name to be assigned to STDENV. It must point to a file that contains the environment variable definitions. Data lines starting with a blank or # will be ignored respectively and not used for setting environment variables. A variable definition must be placed in one line and cannot be continued in the next line.

**STDIN**

This optional parameter is used to specify a path for STDIN.

**pathin**

The path name that is to be assigned to STDIN.

**STDOUT**

This optional parameter is used to specify a path for STDOUT.

**pathout**

The path name that is to be assigned to STDOUT.

**STDERR**

This optional parameter is used to specify a path for STDERR.

*patherr* The path name that is to be assigned to STDERR.

#### UNIX\_command

This is the z/OS UNIX command that is issued under the user ID of the resource that this command belongs to. It is not possible to issue commands for other user IDs. It can be any z/OS UNIX command or the name of a shell script (both fully qualified). The resource that issues this command must have an application type USS.

## Return Codes

**0** Command Issued

**4** Resource is not type USS

**8** OMVS is inactive

**12** Command not issued; failure in INGPYXSR

**16** Command not issued; parameter(s) wrong/missing

#### 1nnn

Command not issued; failure in USS INGCCMD routine, where nnn is the RC received from INGCCMD.

## Restrictions and Limitations

The INGUSS command can be called only by another automation procedure or by a command processor. The AOCQRY command must be invoked first to set the necessary task global variables.

**Note:** The INGUSS command can only be used if the primary JES is available. Therefore, z/OS UNIX resources that use 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. This is because SA z/OS does not create an address space without JES.

## Usage

The following list provides details of some of the variables that can be used to obtain resource data if INGUSS is issued from the automation policy (see “Task Global Variables” on page 44 for a complete list of task global variables that are provided by AOCQRY):

#### &SUBSPATH

The path statement of the resource. The resource must be a process.

#### &SUBSFILE

The filename of the resource. The resource must be a file.

#### &SUBSPID

The ID for the USS process. See also “%PID%” on page 153. &SUBSPID is the process ID returned from the host service BPX1SPN while %PID% is the process ID that is returned from the USS call getpsent().

IBM recommends the use of &SUBSPID in preference to %PID% because problems can arise retrieving the PID in an environment where there are multiple uid 0 users active.

#### &SUBSPORT

The port number of the resource. The resource must be a port.

**&SUBSUSSJOB**

The job name assigned to a process. The resource must be a process.

**&SUBSAPPL**

The application name.

**&SUBSASID**

The address space ID of the address space the process runs in. The resource must be a process.

The information for **&SUBSUSSJOB** and **&SUBSASID** is refreshed with each monitoring cycle. If a process forks and gets a new job name (normally a digit is appended at the end of the original job name), SA z/OS will detect the new job name after the next scheduled monitoring. This works only if SA z/OS internal process monitoring is used.

When the resource becomes inactive, the values of **&SUBSUSSJOB** and **&SUBSASID** are cleared.

In addition, for process resources **%PID%** can be used to get the PID of a process. The command `INGUSS /bin/kill %PID%` results in determining the PID of the process defined by the path of the resource and replacing **%PID%** by the real value of the process ID.

When issuing a command, SA z/OS switches to the user's home directory and sets the following environment variables for the user that the resource belongs to:

- HOME
- USER
- SHELL

The login shell uses these environment variables to detect which UNIX profiles to execute. If the started program should get the whole environment of the user as if this user was logged on, you must use a login shell as the start command.

**Recommendation:**

When using INGUSS to start applications, IBM recommends that you use the **JOBNAME** parameter in order to get a unique job name. For example:

```
INGUSS JOBNAME=&SUBSJOB UNIX_start_command
```

Otherwise, all applications started by SA z/OS without this parameter will have the same job name of INGCUNIX (if the application itself does not change the job name).

If the job name is not unique, specify job type MVS.

**Examples**

1. To start inetd through a login shell, issue the following command:

```
INGUSS JOBNAME=INETD /bin/sh -L -c '/usr/sbin/inetd /etc/inetd.conf'
```

where:

**JOBNAME=INETD**

This is optional. It assigns the MVS job name 'INETD' to the started process.

## INGUSS

**/bin/sh**

The shell.

**-L** The option for the login shell.

**-c** The option to the shell to execute the following command.

**'/usr/sbin/inetd /etc/inetd.conf'**

This is the command that is executed by the login shell

- To start inetd through a login shell, issue the following command:

```
INGUSS JOBNAME=INETD /bin/sh -L -c '/usr/sbin/inetd /etc/inetd.conf
>/tmp/inetd.out 2>/tmp/inetd.err'
```

where:

**>/tmp/inetd.out**

This redirects the command output to /tmp/inetd.out rather than /dev/null.

**2>/tmp/inetd.err**

This redirects the error output to /tmp/inetd.err rather than /dev/null.

- To start cron through a login shell, issue the following command:

```
INGUSS JOBNAME=CRON /bin/sh -L -c '/usr/sbin/cron </tmp/cron.in
>/tmp/cron.out 2>/tmp/cron.err'
```

where:

**</tmp/cron.in**

This redirects the command input to /tmp/cron.in rather than /dev/null.

**>/tmp/cron.out**

This redirects the command output to /tmp/cron.out rather than /dev/null.

**2>/tmp/cron.err**

This redirects the error output to /tmp/cron.err rather than /dev/null.

In the example above the redirection is necessary. If not specified, cron will not hold the pid lock file, and thus multiple pid processes could be started.

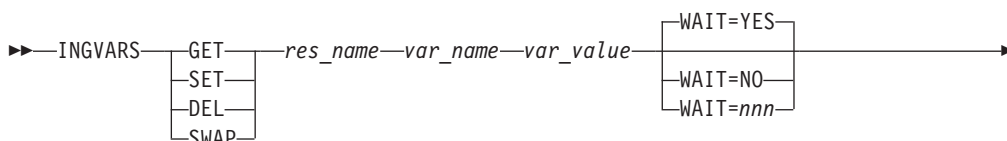
---

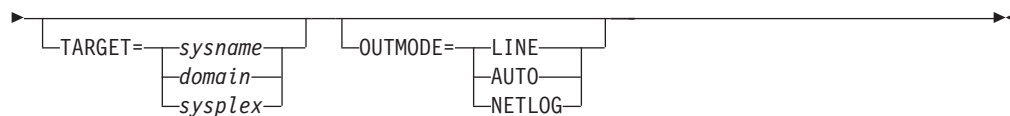
## INGVARS

### Purpose

The INGVARS command is the interface for you to either get or set a shared variable. The shared variable can be associated with an application resource, a system or the sysplex itself.

### Syntax





## Parameters

### GET

Obtains the shared variables.

### SET

Sets the shared variable. Passing a null string resets the variable.

### DEL

Deletes the variable.

### SWAP

Replaces the current setting of the user variable only if the “old” value matches the current setting.

In this case, the specified variable value must consist of two pieces separated by a delimiter. The first piece represents the “old” value while the 2nd piece is the “new” value.

The first character of the variable value is considered to be the delimiter. It can be any printable character, for example /value1/value2.

**Note:** The following restrictions apply when using the SWAP function:

1. Although the specified resource name can contain a wildcard, only 1 resource is allowed.
2. Although the specified variable name can contain a wildcard, only 1 variable name can be specified.
3. Before making the comparison, leading and trailing blanks will be removed.

### *res\_name*

The name of the resource in automation manager format, for example, TSO/APL/AOC8. A wildcard can be specified.

### *var\_name*

The name of the variable. Maximum length is 32 bytes. Can be a wildcard, for example, abc\*, \*abc or \*abc\*. The variable name cannot contain a comma.

### *var\_value*

The value of the variable. Only applicable for the SET function. The value can contain embedded blanks or a keyword/value pair. The value is stored in character format.

### WAIT

Specifies whether to wait until the automation manager has processed the request.

#### YES

Wait for completion. This is the default.

**NO** This causes the subscription request to be sent to the automation manager *without* waiting for its completion.

#### *nnn*

This is the number of seconds to wait before giving up and reporting that a timeout has occurred. The maximum time interval is 999 seconds. If omitted, the time interval is 30 seconds.

### TARGET

For information on the TARGET parameter, refer to *IBM Tivoli System Automation for z/OS Operator's Commands*.

### OUTMODE

For information on the OUTMODE parameter, refer to *IBM Tivoli System Automation for z/OS Operator's Commands*.

Assigning shared variables to resources provides an automatic cleanup of the shared variables. If the resource that the shared variable is associated with is removed (for example, due to an INGAMS refresh), the shared variables are automatically removed as well.

The automation manager provides the following "anchor points" for a shared variable:

#### Application resource

TSO/APL/sysname - this can also be a group resource, for example, CICS/APG

#### System

Resource sysname/SYS/sysname

#### Sysplex

Resource SYSPLEX/GRP

## Return Codes

- 0 OK, continue.
- 1 An error occurred.
- 16 The "old" value does not match the current value of the user variable.

## Restrictions and Limitations

None.

## Usage

Because the automation manager has knowledge of all resources in the sysplex and the automation manager object structures are maintained in a persistent manner, it provides an excellent base for shared variable support.

The automation manager is thus used to manage shared variables. These variables are persistent across automation manager sessions and takeovers and are stored in the takeover file (VSAM). Only when doing a cold start are the shared variables wiped out.

## Examples

### Line-mode Output

Figure 8 on page 157 shows the result of the GET function. The first column is the resource name, the second column is the variable name and the third column is the value of the shared variable.



```

>> ingvars get child* don* outmode=line
CHILD11/APL/AOC8 DONALD BOEBLINGEN
CHILD31/APL/AOC8 DONALD SMITH
CHILD31/APL/AOC8 DON STUTTGARTERSTR
*** End of Display ***

```

Figure 8. INGVARS Command Line-Mode Output

## INGVSTOP

### Purpose

The INGVSTOP command allows an automation procedure to stop a virtual operator station task (VOST).

### Syntax



### Parameters

#### DETACH

Specifies that the VOST is to be stopped with the DETACH command.

#### FORCE

Specifies that the VOST is to be stopped with the STOP FORCE command.

**TASK** Specifies that the VOST is to be stopped with the STOP TASK command.

#### IMMED

Specifies that the VOST is to be stopped with the STOP IMMED command.

#### UNCOND

Specifies that the VOST is to be stopped with the STOP UNCOND command.

### Restrictions and Limitations

The INGVSTOP command can be called only by another automation procedure or by a command processor. The AOCQRY command must be invoked first to set the necessary task global variables. In particular, the SUBSJOB variable must be set. Its content is used as the attach name of the VOST.

Note that the VOST may be still active for a certain time after INGVSTOP has ended with RC=0.

### Return Codes

- 0 Stopping of the VOST was initiated successfully.
- 4 Invalid parameters were specified.
- 6 Environment check failed.
- 8 DETACH or STOP command failed.

## INGVSTOP

### Messages

The following messages are issued by INGVSTOP:

```
AOF010I WRONG NUMBER OF PARAMETERS ENTERED
ING153I command OF name SUCCESSFUL
ING154I command OF name FAILED WITH RC=rc
ING155I ENVIRONMENT CHECK FAILED FOR command. REASON=rs
```

### Usage

Use INGVSTOP as a stop command of a NONMVS type APL.

Consider using INGVSTRT as a start command and INGVMON as a monitor routine. (See “INGVSTRT” on page 158 and “INGVMON” on page 185).

**Note:** It is not recommended to use STOP IMMED because the target task may lose storage or other resources.

You are strongly urged never to use STOP UNCOND because it destroys important task control information in NetView. You might not be able to restart NetView until the next IPL of MVS.

### Examples

To stop a VOST using the DETACH command, enter the following as a stop command in the VOST management APL:

```
INGVSTOP DETACH
```

---

## INGVSTRT

### Purpose

The INGVSTRT command allows an automation procedure to start a virtual operator station task (VOST).

### Syntax

```
▶▶—INGVSTRT—| mode |—command—▶▶
```

**mode:**

```
| [SYNC,] |—————|
| [ASYNC,]
```

### Parameters

**mode** This is a positional parameter that defines the mode that *command* operates in. It is required only if *command* starts with SYNC or ASYNC. It clarifies whether *command* operates synchronously (the default) or asynchronously (it terminates but leaves the VOST active).

It has the following values, which must be followed by a comma:

**SYNC** Use SYNC as a *positional parameter* if *command* operates synchronously. This is the default.

**ASync**

Use ASync if *command* operates asynchronously, that is, it terminates and leaves the VOST active. This prevents message ING156I from being issued.

*command*

The command, including all parameters, to be executed in the VOST. This can be mixed-case and can also be a REXX CLIST.

**Restrictions and Limitations**

The INGVSTRT command can be called only by another automation procedure or by a command processor. The AOCQRY command must be invoked first to set the necessary task global variables. In particular, the SUBSJOB variable must be set. Its content is used as the attach name of the VOST.

The ATTACH command is used to start the VOST and therefore the restrictions of the attach command also apply to the command that is specified with INGVSTRT.

**Return Codes**

- 0 VOST started successfully.
- 4 Invalid parameters were specified.
- 6 Environment check failed.
- 8 ATTACH command failed.

**Messages**

The following messages are issued by INGVSTRT:

```
AOF010I WRONG NUMBER OF PARAMETERS ENTERED
ING151I ATTACH OF name SUCCESSFUL
ING152I ATTACH OF name FAILED WITH RC=rc
ING155I ENVIRONMENT CHECK FAILED FOR command. REASON=rs
```

**Usage**

Use INGVSTRT as the start command for an APL of type NONMVS.

Consider using INGVSTOP as a stop command and INGVMON as a monitor routine. (See "INGVSTOP" on page 157 and "INGVMON" on page 185).

**Examples**

To start the CLIST myclist in a VOST, create an APL of type NONMVS and enter, for example, the following as a start command:

```
INGVSTRT MYCLIST PARM1,2ND,THIRD
```

Note that parameters can be in mixed case.

**INGVTAM****Purpose**

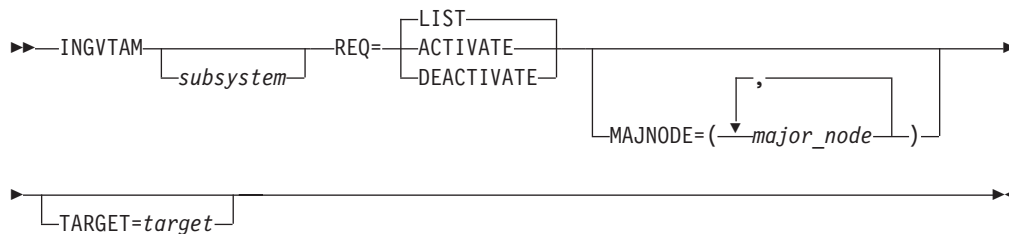
The INGVTAM command lets you:

- Register an application with VTAM application node recovery.
- Issue recovery commands for all applications registered with VTAM application node recovery when VTAM has restarted.
- List applications that are registered for application node recovery.

## INGVTAM

- List major nodes that are in use by applications. When the subsystem terminates, the major nodes are automatically purged.

### Syntax



### Parameters

#### *subsystem*

The subsystem parameter specifies the name of the subsystem that is registering with SA z/OS VTAM application recovery. This parameter is required with REQ=ACTIVATE to register a subsystem. If it is omitted with REQ=ACTIVATE, all subsystems currently registered will have the VTAMUP message command policy driven to allow them to take actions when VTAM is restored to active service. This parameter is required with REQ=DEACTIVATE.

#### REQ

Specifies the request. It can be one of the following:

##### LIST

If no subsystem is specified, it lists all subsystems registered for VTAM application node recovery. If a subsystem is specified, it lists all the major nodes registered for that subsystem.

##### ACTIVATE

If the subsystem parameter is specified, it registers the list of major nodes as specified in the MAJNODE= parameter and issues VTAM ACTIVATE commands for them. If the subsystem parameter is not specified, REQ=ACTIVATE issues the commands in the messages policy VTAMUP for every subsystem that is registered for application node recovery.

##### DEACTIVATE

A subsystem must be specified for this request. This request issues VTAM INACT commands for the major nodes that were previously registered. INACT commands are not issued for any major node that contains model resources or is in use by another registered application.

#### MAJNODE

This defines a list of VTAM application major nodes that will be acted on.

#### TARGET

For information on the TARGET parameter, refer to *IBM Tivoli System Automation for z/OS Operator's Commands*.

### Return Codes

- 0 Normal End.
- 4 Warning (Vary command failed).
- 8 Error.

## Restrictions and Limitations

To use the INGVTAM command SA z/OS must be fully initialized.

## Usage

It is recommended that you issue the REQ=ACTIVATE and REQ=DEACTIVATE commands on the same system as the subsystems concerned. It is recommended that you place REQ=ACTIVATE in the application's PRE-START and ACORESTART policies. However, REQ=DEACTIVATE should be placed in the application's SHUTFINAL policy. For the VTAM subsystem, the INGVTAM REQ=ACTIVATE command should be defined to the UP message policy as a command.

## Examples

If you enter INGVTAM REQ=LIST the output is similar to Figure 9.

```
List of subsystems registered with VTAM
Subsystem Subsystem Subsystem Subsystem
EYUCMS1A
*** End of Display ***
```

Figure 9. INGVTAM REQ=LIST Output

If you enter INGVTAM *subsystem* REQ=LIST the output is similar to Figure 10.

```
List of major nodes registered with subsystem subsys
Major Node Type Major Node Type
KEY1BCPA APPL
*** End of Display ***
```

Figure 10. INGVTAM *subsys* REQ=LIST Output

To register a subsystem for application node recovery, specify, for example:

```
INGVTAM &SUBSAPPL REQ=ACTIVATE MAJNODE=(IPSMBC)
```

To deregister a subsystem for application node recovery, specify, for example:

```
INGVTAM &SUBSAPPL REQ=DEACTIVATE
```

---

## ISSUEACT (ISSUECMD, ISSUEREP)

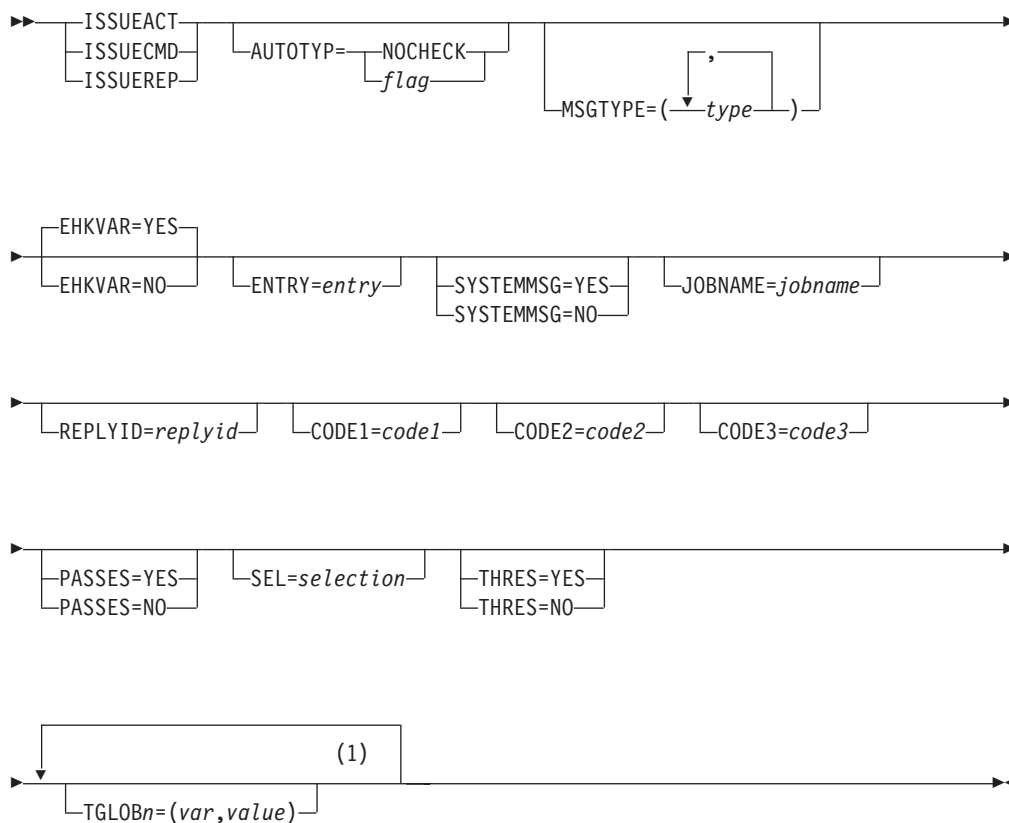
### Purpose

ISSUEACT, ISSUECMD, and ISSUEREP are defined as synonyms for the same command that can be used to trigger your own commands, replies, or both, from messages that are defined in the automation policy item MESSAGES/USER DATA under consideration of the automation flags.

If the command is called as ISSUECMD, only commands are issued, whereas if it is called as ISSUEREP, only replies are issued. When called as ISSUEACT, it issues commands and replies according to the given selection criteria that are passed as parameters.

In addition, the ISSUEACT command includes special message processing for some critical DB2 messages and for JES2 message \$HASP099. For further details see the sections “Critical Event Monitoring” and “JES2 Shutdown Processing” in *IBM Tivoli System Automation for z/OS Customizing and Programming*.

## Syntax



### Notes:

- 1 The variable *n* can be 1–10 (that is, TGLOB1,TGLOB2,...,TGLOB10)

## Parameters

### AUTOTYP

The automation flag that is to be checked. If the flag is turned off no commands or replies are issued.

### NOCHECK

If NOCHECK is specified, the RECOVERY flag is checked, but the commands or replies (or both) are issued regardless of its setting.

### flag

This must be one of the following values:

- AUTOMATION
- INITSTART
- RECOVERY
- RESTART
- START
- TERMINATE

If SYSTEMMSG=YES is specified, NOCHECK, AUTOMATION, and RECOVERY are the only valid values for AUTOTYP.

If no AUTOTYP value is coded and SYSTEMMSG=YES, AUTOTYP defaults to RECOVERY.

If no AUTOTYP value is coded and SYSTEMMSG=NO, the default value is determined according to the following steps:

1. If startup of the application is in progress, AUTOTYP=START
2. If shutdown of the application is in progress, AUTOTYP=TERMINATE
3. If neither a startup nor a shutdown is in progress, a value for AUTOTYP is taken that corresponds to the actual status of the application:

| AUTOTYP   | Actual Status                                                                          |
|-----------|----------------------------------------------------------------------------------------|
| START     | ACTIVE, ENDING, EXTSTART, RESTART, RUNNING, STARTED, STARTED2                          |
| TERMINATE | ABENDING, AUTOTERM, BREAKING, HALFDOWN, STOPPING, STUCK, ZOMBIE                        |
| RECOVERY  | AUTODOWN, BROKEN, CTLDOWN, DOWN, ENDED, FALLBACK, HALTED, INACTIVE, MOVED, STOPPED, UP |

4. If no actual status information is available, RECOVERY is taken as the default value for AUTOTYP

#### MSGTYPE

This value is a list of the message IDs in the MESSAGES/USER DATA policy item where the commands or replies (or both) to be issued are defined. It defaults to the ID of the message that initiated ISSUEACT, ISSUECMD or ISSUEREP, if the command is called from the NetView automation table. If the command is not driven by a message, you must supply this parameter.

The embedding brackets are not needed if only one message ID is specified.

#### EHKVAR

This parameter determines whether the tokens of the parsed message text are to be stored in task global variables EHKVAR0 through EHKVAR9 and EHKVART.

**YES** The tokens of the triggering message are to be assigned to the task global variables EHKVAR $n$ .

**NO** No values are to be assigned to the task global variables EHKVAR $n$ . The values in EHKVAR $n$  remain unchanged.

#### ENTRY

This value is the entry name of the definition in the automation policy where the commands or replies (or both) to be issued are defined.

- If ISSUEACT, ISSUECMD or ISSUEREP is called from the NetView automation table, *entry* defaults to:
  - The application name, as determined from the job name, for application messages
  - The system type (MVSESA) for system messages

Otherwise you must supply this parameter.

#### JOBNAME

This parameter is used to pass the job name when ISSUEACT, ISSUECMD or ISSUEREP is not called from the NetView automation table.

#### REPLYID

This parameter is used to pass the reply ID when the command has to reply to a WTOR, but has not been called from the NetView automation table.

**Note:** This parameter is not valid if the command is called as ISSUECMD.

## ISSUEACT

### SYSTEMMSG

Indicates whether the message is a system message or an application message.

#### YES

The message was issued by a system rather than an application. SYSTEMMSG defaults to YES if no job name can be obtained from the message details, and neither the JOBNAME nor the ENTRY parameter is specified. Furthermore it defaults to YES, if the job name that is obtained is not defined to SA z/OS and if, in addition, no ENTRY parameter is specified or its value is the system type (MVSESA).

**NO** The message was issued by an application that must be defined to SA z/OS.

### CODE1=code1 CODE2=code2 CODE3=code3

When specified, the codes that are passed are used to search for code entries for a particular message ID that is specified in the automation policy MESSAGES/USER DATA.

If the command is called as ISSUEACT or ISSUEREP, and if there are no command or reply entries besides the code definition to the given message ID, the response to the matching entry is used as the reply to a WTOR.

Otherwise the response to the matching entry is used as the option to select the commands or replies (or both) to be issued from the automation control file. If no code match occurs for the specified codes, the value ALWAYS is used to select the commands or replies (or both) to be issued.

A selection string of `"*IGNORE"` that is returned from the code match function is treated as a no-operation instruction. This can help make the CODE definitions in the automation policy simpler because you can filter out the entries that no processing should be done for by SA z/OS.

The CODE parameters are mutually exclusive to the PASSES=YES and SEL parameters.

### MSG

This parameter is used to pass an alternate message text when ISSUEREP or ISSUEACT is triggered by a WTOR. The value of this parameter is used as the message text instead of the message text of the triggering message to be forwarded to SDF and NMC.

This parameter is rejected if the command is called as ISSUECMD.

### PASSES

Specifies whether passes are used to issue the commands or replies.

#### YES

Passes are used to issue the commands or replies. The pass count is incremented only if the automation flag is turned on. The pass count is keyed by message ID, and for normal messages the count is reset when the application is shut down. For system messages, the pass count is reset when NetView is recycled.

This value is mutually exclusive to the CODE parameters.

**NO** Passes are not used to issue the commands or replies.

If PASSES is not coded, it defaults to YES if the AUTOTYP parameter has a value other than START or TERMINATE, and the command or reply entries of the specified ENTRY and MSGTYPE in the automation control file use pass selection options. When START or TERMINATE is the value for the AUTOTYP parameter, YES is only assumed as the default value for



the PASSES parameter if there are no command or reply entries defined with a selection name other than *PASS<sub>nn</sub>*. In all other cases, the default value to PASSES is set to NO.

#### SEL

Specifies a selection string that is to be used to determine the commands or replies that are to be issued, along with all commands or replies defined without a selection value.

This parameter is mutually exclusive to the CODE parameters.

#### THRES

Specifies whether defined thresholds for the minor resource *entry.type* are checked before issuing commands or replies.

**YES** Thresholds are checked before issuing commands or replies.

An error record for minor resource *entry.type* is written to the automation status file and the frequency of the written error records is compared with the defined threshold levels for this resource.

As long as no option has been derived from other criteria (such as the start or stop type, the PASSES parameter or CODE parameters), the name of the exceeded threshold level (ALWAYS, INFR, FREQ, or CRIT) is used to select defined commands or replies with these selection options. If no commands or replies with these selection options are defined, all commands or replies defined for the given entry and type are issued if the critical threshold has not been exceeded.

If a selection option has already been provided by other criteria to select commands and replies, these commands or replies are only issued if the critical threshold has not been exceeded.

**NO** Thresholds are not checked before issuing commands or replies.

If THRES is not coded, its value defaults to YES if there are thresholds defined for the minor resource *entry.type*. Otherwise the value of THRES is assumed to be NO.

#### TGLOB<sub>n</sub>

This parameter instructs ISSUEACT to store a certain value in a given task global variable. You can specify up to 10 variables.

##### *var*

The name of the task global variable that is to be set.

##### *value*

The value that is to be stored in the task global variable

**Note:** Be careful not to specify the names of other task global variables (for example, EHKVAR<sub>n</sub>) because they will be overwritten.

## Restrictions and Limitations

- ISSUEACT, ISSUECMD and ISSUEREP will only work when SA z/OS is fully initialized.
- SYSTEMMSG=YES is only accepted if no job name is provided by the JOBNAME parameter and no ENTRY parameter is specified or the value of the common global variable AOFSYSTEM is passed as the value for it.
- SYSTEMMSG=YES is only valid in combination with the AUTOTYP values NOCHECK, RECOVERY, or AUTOMATION.

## ISSUEACT

- If ISSUEACT, ISSUECMD or ISSUEREP is driven by a delete operator message, no commands or replies are issued that are driven by such a message.
- The command must run on the working operator task of the application.

### Usage

You should normally call the ISSUEACT command from the NetView automation table.

The triggering message of the ISSUEACT command is stored in the SAFE called AOFMSAFE. All commands that are triggered through ISSUEACT and that are executed on the task that is currently executing ISSUEACT have access to this SAFE.

Do not call OUTREP in addition to the ISSUEACT command for a triggering WTOR. If the triggering WTOR is not replied to in ISSUEACT (or ISSUECMD or ISSUEREP), OUTREP is automatically called to record the WTOR.

If AUTOTYP=START is flagged and you specify PASSES=NO and no CODE parameters, the current start type is taken as the selection for the commands or replies to be issued. If no start type is provided, NORM is assumed as the default start type.

If AUTOTYP=TERMINATE is flagged and you specify PASSES=NO and no CODE parameters, the current stop type will be taken as the selection for the commands or replies to be issued. If no stop type is provided, NORM is assumed as the default stop type.

If no selection option has been derived from criteria such as start or stop type, PASSES parameter, CODE parameters or the result of threshold checking, ALWAYS is assumed as the default option for selecting defined commands or replies to be issued.

If the value of the advanced automation option AOFSTATUSCMDSEL is set to zero, all defined commands or replies for a specified status as message ID are issued, regardless of any defined selection option. That is, no option from criteria such as start or stop type, PASSES parameter, CODE parameters or the results of threshold checking is used to select defined commands or replies. No minor resource threshold checking is done in this case.

### Task Global Variables

#### **EHKVAR0 through EHKVAR9 and EHKVART**

When defining the commands or replies in the automation policy that are to be issued by this command, the variables &EHKVAR0 through &EHKVAR9 and &EHKVART can be used to be substituted by the tokens of the parsed message that has driven this command. &EHKVAR0 will be substituted by the message ID, &EHKVAR1 by the first token of the message text after the message ID, &EHKVAR2 with the second token and so forth. &EHKVART will be substituted by the trailing message text after the 9th token.

### Examples

#### **Example 1**

This example shows an automation procedure that calls the ISSUEACT command to handle the HSM subsystem message, ARC0027I.

The automation policy is as follows:

```

AOFK3D0X SA z/OS - Command Response Line 1 of 4
Domain ID = IPSNO ----- DISPACF ----- Date = 07/19/00
Operator ID = SAUSER Time = 18:20:45

Command = ACF ENTRY=HSM,TYPE=ARC0027I,REQ=DISP
SYSTEM = KEY3 AUTOMATION CONFIGURATION DISPLAY - ENTRY= HSM

AUTOMATION CONFIGURATION DISPLAY - ENTRY= HSM
TYPE IS ARC0027I
CMD = (,,'MVS S HSMPL0GB')
END OF MULTI-LINE MESSAGE GROUP

```

Figure 11. DISPACF Sample Panel

The NetView automation table entry to call ISSUEACT is:

```

IF MSGID = 'ARC0027I' THEN
EXEC(CMD('ISSUEACT') ROUTE(ONE %AOFOPGSSOPER%));

```

The automation flag to check depends on the phase in the life cycle of the HSM subsystem. If no start up or shutdown is in progress for the application, ISSUEACT checks the recovery flag to validate that automation is allowed before issuing the command. If automation is allowed and message ARC0027I is received for job DFHSM, relating to the HSM subsystem, a command is issued that saves the HSM data set. If message ARC0027I is received for any job other than DFHSM, the message is not automated.

If you specify a clist named MYCLIST instead of an MVS command for the message ARC0027I in the message policy of the customization dialog, this clist can access the original message that triggered ISSUEACT via the named safe AOFMSAFE. Thus you are able to access the message attributes and all lines of a multiline message. The code to access this safe should look similar to the following:

```

/* MYCLIST */

...

/* Get the message from the SAFE called AOFMSAFE */
"PIPE (STAGESEP | NAME GETMSG)" ,
 "SAFE AOFMSAFE" ,
 "| STEM orig_msg."

...

Exit

```

## Example 2

This example shows how ISSUEACT can be used to automatically respond to WTOR AHL125A, which is issued by GTF during initialization and which allows SA z/OS to accept or reject the trace options that GTF will use.

To enable SA z/OS to automatically accept the trace options, define value U as the reply to message AHL125A. To do this, select the MESSAGES/USER DATA policy item from the Policy Selection panel for the GTF subsystem in the customization dialog. In the Message Processing panel, specify AHL125A as the message ID and call action REP to get the related Reply Processing panel. Specify U in the **Reply Text** field.

## ISSUEACT

Return to the Message Processing panel.

When you call action OVR, you can see that the automation table entry that will be created during the build process for the automation policy:

```
IF MSGID = 'AHL125A' THEN
EXEC(CMD('ISSUEACT') ROUTE(ONE %AOFPGSSOPER%));
```

ISSUEACT is called without parameters. Therefore the automation flag to be checked depends on the phase in the lifecycle of the GTF subsystem. Because message AHL125A is issued during the initialization of GTF, ISSUEACT checks the start flag to validate that automation is allowed before issuing the reply.

If automation is allowed and message AHL125A is received for job GTFPROD that is related to the GTF subsystem, ISSUEACT replies with value U to accept the trace options and to continue its initialization. If message AHL125A is received for any job other than GTFPROD, the message is not automated.

---

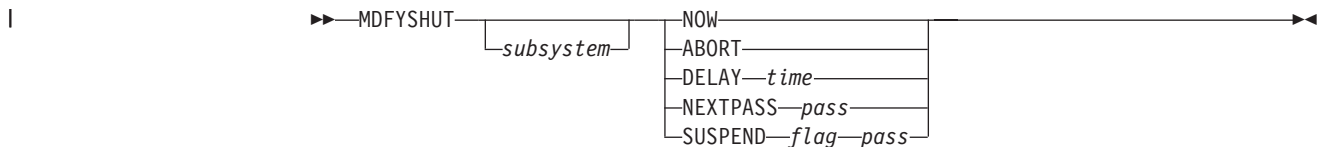
## MDFYSHUT

### Purpose

The MDFYSHUT command sets the AOFSHUTMOD task global variable to whatever value is contained in the MDFYSHUT parameter string. The AOFSHUTMOD value is then used by the shutdown program.

MDFYSHUT also provides support for a SUSPEND function.

### Syntax



### Parameters

#### Subsystem

Specifies the subsystem involved in the shutdown. If omitted, the SUBSAPPL task global variable is used to identify the subsystem.

#### NOW

The next shutdown pass will occur as soon as possible.

#### ABORT

Stops the shutdown process bringing the subsystem in HALFDOWN status.

#### DELAY

The next shutdown pass that will occur after time instead of the shut delay defined for the subsystem. Time must be in the hh:mm:ss format. If only a 2-digit value is specified for time, SA z/OS assumes it to be a value for minutes. If only a 2-digit value preceded by a colon is specified for time, SA z/OS assumes it to be a value for seconds.

#### NEXTPASS

The next shutdown pass that will be processed (after the subsystem shut delay) is the pass value (2nd parameter), not the current pass plus one.

**SUSPEND**

Determines how the shutdown is suspended, where:

*flag*

Is the name of a common global variable that is used to determine how the shutdown is suspended. If the flag is set off (meaning its value is 0), the shutdown will be stopped and the flag will be checked again on the next pass. If the flag is set on (meaning its value is 1) the shutdown will continue, that is, it is no longer suspended. After stopping, the value of the common global variable is set back to 0.

*pass*

Is the number of the pass that the MDFYSHUT SUSPEND command is coded on. It must be included so that MDFYSHUT can return to this pass and recheck the flag.

**Note:** The SUSPEND parameter can only be used in the Shutdown definitions of the automation policy.

**Restrictions and Limitations**

The MDFYSHUT command can be used on any pass of the shutdown.

The routine that contains MDFYSHUT must run on the default task, that is, leave the task field blank.

The routine that contains MDFYSHUT cannot be rescheduled with a CMD LOW.

---

**OUTREP****Purpose**

The OUTREP command captures and saves MVS reply identifiers for applications that issue outstanding replies. Some applications issue an outstanding reply when they start, and that reply is used for critical operator communication or shutdown commands. This command captures these reply IDs and their message text and saves them in case the automation code needs them for recovery or shutdown.

Typically, OUTREP is called from the NetView automation table.

**Syntax**

```

▶▶—OUTREP—┌──────────┴──────────▶▶
 └──message──┘

```

**Parameters***message*

The message text for the outstanding reply. If not specified it will be picked up from the default safe.

**Restrictions and Limitations**

If another command such as ISSUEACT/ISSUEREP/ISSUECMD, ACTIVMSG, HALTMSG, TERMMSG or INGMON is called for the WTOR, you should not code an additional call to OUTREP for the same WTOR. If the command cannot find a

## OUTREP

value to reply to the WTOR with, it automatically calls OUTREP to record the WTOR. This also happens if the command is called and finds that the automation for the message is turned off.

### Usage

You should normally call the OUTREP command from the NetView automation table.

The OUTREP command attempts to determine the application name from the job name that is associated with the message. It then calls CDEMATCH with:

```
CODE1=msgid
CODE2=jobname
```

to determine what is to be done with the outstanding WTOR.

If an application is found, CDEMATCH searches the Automation Control File for CODE entries that are associated with ENTRY-TYPE pairs of *application*-WTORS where *application* is the application name as determined from the job name.

If an application cannot be found, or there is no match from the first search, CDEMATCH searches CODE entries that are associated with ENTRY-TYPE keys of MVSESA-WTORS.

If a successful match occurs, CDEMATCH returns a value consisting of two words that instruct OUTREP what to do with the WTOR:

- **First word:** Assigns the severity that determines the color of the WTOR in SDF and NMC.
- **Second word:** Assigns the priority. WTORS with a priority of PRIMARY are used by SA z/OS as outstanding WTORS but those with a priority of SECONDARY are not.

The following table shows the valid values for the severity with the resulting status in SDF and the colors of the WTORS in SDF and NMC.

| Severity  | Status in SDF | Default Color in SDF | Color in NMC |
|-----------|---------------|----------------------|--------------|
| NORMAL    | NWTOR         | Green                | Green        |
| UNUSUAL   | UWTOR         | Yellow               | White        |
| IMPORTANT | IWTOR         | Pink                 | Pink         |
| CRITICAL  | CWTOR         | Red                  | Red          |
| IGNORE    | -             | -                    | -            |

Any definite abbreviation can be used to specify the severity.

By default an incoming WTOR is considered to be of priority PRIMARY with a severity of UNUSUAL. This also means that any code definitions where you have entered incorrect data in the **Value Returned** field default to UNUSUAL PRI.

The codes that CDEMATCH is to search on are entered against a message ID of WTORS in the Code Processing panels of the customization dialogs. Figure 12 on page 171 shows an example of code definitions to the message ID WTORS that are specified at the entry of the NetView application with job name NETVAPPL.

| Code 1  | Code 2 | Code 3 | Value Returned |
|---------|--------|--------|----------------|
| DSI802A | *      |        | NORMAL PRI     |
| DSI803A | *      |        | NORMAL PRI     |
| TEST001 | *      |        | NORMAL SEC     |

Figure 12. Code Processing Panel for an Application Resource

Figure 13 shows an example of code entries for the MVSESA resource.

| Code 1  | Code 2 | Code 3 | Value Returned |
|---------|--------|--------|----------------|
| IEA793A | *      |        | IMPORTANT SEC  |
| IEC507D | *      |        | NORMAL SEC     |
| IEF235D | *      |        | NORMAL SEC     |
| IEF238D | *      |        | IMPORTANT SEC  |
| IEF455D | *      |        | NORMAL SEC     |
| IEF458D | *      |        | NORMAL SEC     |

Figure 13. Code Processing Panel for the MVSESA Resource

These code entries result in the following behavior:

- If the job NETVAPPL issues one of the messages DSI802A or DSI803A, it is assigned a severity of NORMAL and is displayed in the related color in SDF and NMC. SA z/OS can use this outstanding reply, for example, to shut down job NETVAPPL.
- If any Clist running in NetView with job name NETVAPPL issues a WTOR with message ID TEST001, it is also assigned a severity of NORMAL and is displayed in the relating color in SDF and NMC, but it cannot be used to shut down this NetView.
- If one of the messages defined in the code entries to the MVSESA resource in Figure 13 is issued by any application or MVS component, and no replies are defined in SA z/OS to be issued in response to them, these messages are stored as secondary WTORs and are displayed in SDF and NMC with the specified severity.

## Task Global Variables

None.

## Examples

The following is an example of calling the OUTREP command directly from the NetView automation table:

```
IF MSGID='DSI802A' & DOMAINID = %AOFDOM%
THEN
EXEC(CMD('OUTREP') ROUTE(ONE %AOFOPSYSOPER%));
```

In this example, OUTREP is called for the NetView outstanding reply message, DSI802A. %AOFDOM% is a synonym defined to be the current domain. %AOFOPSYSOPER% is a cascade for processing WTORs. Both are defined in AOFMSGSY.

---

## TERMMMSG

### Purpose

You can use the TERMMMSG command to respond to the termination message of an application by changing the SA z/OS status of the application. TERMMMSG calls the ISSUEACT command to also issue commands and replies that are defined in

## TERMMSG

the automation policy for the ID of the termination message and for the new status. Typically, TERMMSG is called from the NetView automation table.

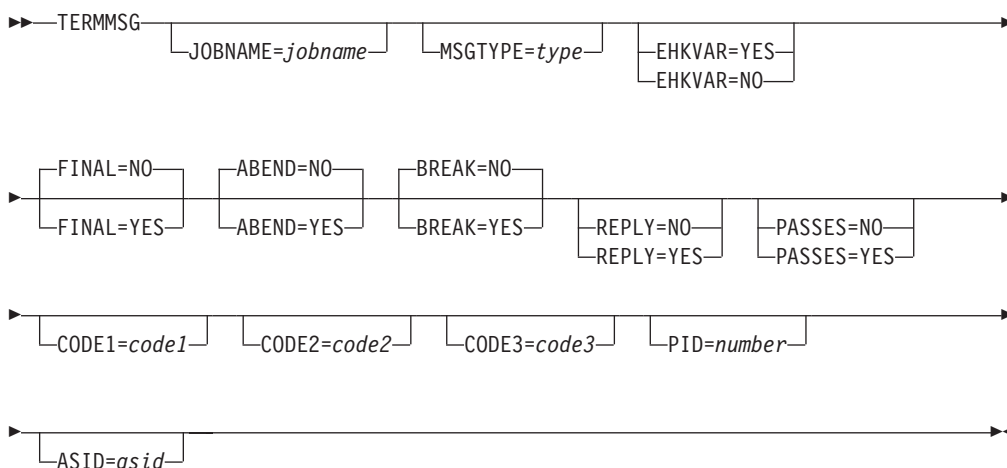
The status that the application is placed in by TERMMSG depends on a number of conditions, including the values of the FINAL, ABEND, and BREAK parameters. The values of the FINAL, ABEND and BREAK parameters may in turn depend on the values of the CODE parameters. The following table shows the statuses that TERMMSG may place an application in.

Table 8. TERMMSG Status Transitions

| Status              | Description                                                                   | Final | Abend | Break |
|---------------------|-------------------------------------------------------------------------------|-------|-------|-------|
| STOPPING            | Application terminated externally                                             | N     | N     | N     |
| ENDING              | For transient applications                                                    | N     | N     | N     |
| ABENDING            | Application abend                                                             | N     | Y     | N     |
| BREAKING            | Non-recoverable abend                                                         | N     | N     | Y     |
| STOPPED             | Application shutdown externally                                               | Y     | N     | N     |
| ENDED               | Transient application shutdown                                                | Y     | N     | N     |
| BROKEN              | Non-recoverable abend                                                         | Y     | N     | Y     |
| RESTART             | Restart after abend                                                           | Y     | Y     | N     |
| AUTOTERM            | No change during shutdown                                                     | N     | N     | N     |
| AUTODOWN or RESTART | System is being shut down. The status will depend on the shutdown parameters. | Y     | ?     | ?     |
| ZOMBIE              | Occurs if there are problems with the address space cleanup.                  | Y     | ?     | ?     |

For information about how the CODE parameters can affect the values of FINAL, ABEND, and BREAK see the description of “The CODE Parameter” on page 174.

## Syntax





## Parameters

### JOBNAME

The name of the job that the message is for. If not specified, the job name is taken from the message's job name field. You must supply a value for the job name if you are calling TERMMMSG from a CLIST.

### MSGTYPE

This parameter is used to search for command and reply entries to *subsystem/msgtype*-pairs in the automation control file, where *subsystem* is the subsystem name derived from the job name.

When a match occurs, the commands associated with the entries are issued.

If the MSGTYPE parameter is not specified, the message identifier of the message that TERMMMSG is called for is taken as the default.

### EHKVAR

This parameter determines whether the tokens of the parsed message text are to be stored in task global variables EHKVAR0 through EHKVAR9 and EHKVART.

#### YES

The tokens of the triggering message are to be assigned to the task global variables EHKVAR $n$ .

**NO** No values are to be assigned to the task global variables EHKVAR $n$ .

### FINAL

Indicates whether this is the final termination message. If no FINAL value is coded, TERMMMSG defaults to FINAL=NO.

#### YES

The message is the final termination message for the application. The application will be placed into the appropriate status, depending on the values of the ABEND and BREAK parameters. See Table 8 on page 172 for details. If it is monitorable, the application is not placed into a down status until an application monitor check confirms that it has left the machine. If it is not monitorable, the application is placed into a down status after its termination delay time.

**NO** This is not the final termination message.

### ABEND

Indicates whether the application is suffering a recoverable abend. If no ABEND value is coded, TERMMMSG defaults to ABEND=NO.

#### YES

The application is suffering a recoverable abend. The application will be placed into the appropriate status, depending on the value of the FINAL parameter. See Table 8 on page 172 for details.

When the final termination message for an abending application (FINAL=YES) is received, the error threshold is checked and the application is restarted if it has not exceeded its critical error threshold.

**NO** The application is not suffering a recoverable abend.

### BREAK

Indicates whether the application is suffering a non-recoverable abend. The application will be placed into the appropriate status, depending on the value of the FINAL parameter. If no BREAK value is coded, TERMMMSG defaults to BREAK=NO.

## TERMMSG

### YES

The application is suffering a non-recoverable abend and should be placed into BREAKING status. When its final termination message is received (FINAL=YES) it is placed into BROKEN status. SA z/OS will not restart it from this status without human intervention through the SETSTATE command dialog.

**NO** The application is not suffering a non-recoverable abend.

### REPLY

This parameter determines whether a defined reply is issued for a message that TERMMSG has been called for.

### YES

A defined reply in the automation policy for the message that is being handled by TERMMSG is issued. REPLY=YES is assumed as the default if the message is a WTOR, otherwise the default is REPLY=NO.

**NO** A defined reply for a WTOR that is being handled by TERMMSG is not issued.

### PASSES

Specifies whether passes are used to issue commands or replies (or both) that have been defined in the automation policy.

### YES

PASSES=YES is passed to the ISSUEACT command.

**NO** PASSES=NO is passed to the ISSUEACT command.

**CODE1=code1**

**CODE2=code2**

**CODE3=code3**

When specified, the codes that are passed are used to search for code definitions for the termination message in the automation policy MESSAGES/USER DATA. First the automation policy is searched for code definitions against the message ID of the subsystem that issued the termination message. If these cannot be found, the automation policy is searched for code definitions against the message ID MVSESA.

The meaning of the codes depends on the NetView automation table entry that invoked TERMMSG.

The value returned for the matching code definition can consist of two tokens. The first token is used as the action to modify the FINAL, ABEND and BREAK parameters of TERMMSG in the following way:

| Action   | Final | Abend | Break |
|----------|-------|-------|-------|
| STOPPING | -     | -     | -     |
| STOPPED  | Yes   | -     | -     |
| ABENDING | -     | Yes   | -     |
| ABENDED  | Yes   | Yes   | -     |
| BREAKING | -     | -     | Yes   |
| BROKEN   | Yes   | -     | Yes   |
| IGNORE   | -     | -     | -     |

If IGNORE is returned as the first token, the processing of TERMMMSG stops. In this case the second token of the returned value is not considered. The status of the application is not updated and no command or reply is issued by TERMMMSG.

If specified, the second token of the returned value is used as the start type for the subsystem's next startup. The next start type is set with the INGSET command.

TERMMMSG does not apply the code values for selecting defined commands or replies to be issued.

**Note:** An action of IGNORE can be used for messages not resulting in the termination of the application. TERMMMSG will not perform any status change and will simply stop processing.

#### PID

The process ID of the resource. Together with the ASID, it uniquely identifies the resource.

#### ASID

The ASID that is associated with the resource. Together with the PID, it uniquely identifies the resource.

## Restrictions and Limitations

- If TERMMMSG is driven by a delete operator message, no action is taken in response to this message.
- If a normal termination message (ABEND=NO,BREAK=NO) is received for an application that is not being shut down by SA z/OS (and is already in the AUTOTERM status), it is placed into the STOPPING status. When its final termination message has been processed, its Restart option is checked. If this is ALWAYS it is placed into the RESTART status. If the Restart option is not ALWAYS it is placed into the STOPPED status.

This behavior can be changed using the AOFRESTARTALWAYS advanced automation option.

- Once an application has entered a serious error condition (a status of AUTOTERM, STOPPING, ABENDING, or BREAKING), termination messages indicating less important error conditions are ignored.
- Commands for a status are only issued the first time the status is entered.
- If the TERMMMSG command is called on a task other than the AOFWRKxx auto operator that is responsible for the subsystem, TERMMMSG will schedule itself to that AOFWRKxx auto operator. That is, when the calling procedure regains control, the status of the subsystem may not yet have changed.
- Only termination messages for applications with known address space IDs are processed by TERMMMSG.

The address space ID is not checked if TERMMMSG is called from an automation procedure (CLIST), or if TERMMMSG has been triggered by message BPXF024I.

The address space ID is also ignored if the job name parameter that was specified differs from the job name associated with the triggering message.

## Usage

The definition of termination messages ensures early detection of any problems with subsystems. A number of termination messages is already known to SA z/OS. **You can define an additional termination message using the MESSAGES/USER DATA policy item of the application to set the AT status of**

**the message as terminating or terminated.** During the automation policy build an appropriate NetView automation table statement is created that calls TERMMSG. See the MESSAGES/USER DATA policy item in *IBM Tivoli System Automation for z/OS Defining Automation Policy* for more details about defining termination messages.

Message IEF404I is used by SA z/OS as the final termination message for all applications. The following example shows how TERMMSG is called by IEF404I in the automation table:

```
IF MSGID='IEF404I' & TOKEN(2) = SVJOB & DOMAINID=%AOFDOM%
 & ATF('ING$QRY APPL,,JOB='VALUE(SVJOB)) ^= ''
THEN
EXEC(CMD('TERMMSG FINAL=YES,JOBNAME=' SVJOB) ROUTE(ONE %AOFPGSSOPER%));
```

The ING\$QRY NetView automation table function is used to screen the message before invoking TERMMSG. See Chapter 4, “ING\$QRY NetView Automation Table Function,” on page 187 for more information.

Using code definitions to a message avoids having to code multiple automation table statements or to issue multiple commands to call TERMMSG.

The following example shows how TERMMSG is called by generic message IEF450I:

```
IF MSGID='IEF450I' & TOKEN(2) = SVJOB & DOMAINID=%AOFDOM%
 & ATF('ING$QRY APPL,,JOB='VALUE(SVJOB)) ^= ''
 & TEXT = . 'ABEND=' SCODE UCODE .
THEN
EXEC(CMD('TERMMSG JOBNAME='SVJOB ',CODE1=' SVJOB ',CODE2='
SCODE ',CODE3=' UCODE) ROUTE(ONE %AOFPGSSOPER%));
```

If you are calling TERMMSG from an automation procedure, and this calling procedure is not running on the AOFWRKxx automation operator that is responsible for the affected subsystem, the TERMMSG command will be routed to that operator. The TERMMSG command will run asynchronously to the calling procedure. This means that when the calling procedure regains control, the status of the affected subsystem may not yet have changed.

All commands and replies that are triggered through TERMMSG have access to the SAFE, called AOFMSAFE, that stores the message that caused the TERMMSG call.

## Task Global Variables

### EHKVAR0 through EHKVAR9 and EHKVART

When defining the commands in the automation control file to be issued by TERMMSG command, the variables &EHKVAR0 through &EHKVAR9 and &EHKVART can be used to be substituted by the tokens of the parsed message that has driven TERMMSG. &EHKVAR0 will be substituted by the message ID, &EHKVAR1 will be substituted by the first token of the message text after the message ID, &EHKVAR2 with the second token and so on. &EHKVART will be substituted by the trailing message text after the 9th token.

## Examples

TERMMSG is called with CODE1=ABENDED and CODE2=S222 by the termination message of an application that has the following codes defined for it:

| Code 1 | Code 2 | Code 3 | Value Returned |
|--------|--------|--------|----------------|
| ERROR* | \$PJF  |        | STOPPING       |
| ABEND* | S222   |        | ABENDING HOT   |

A match occurs with the second code definition, and the application is placed in the status ABENDING and the start type for the next application startup is set to HOT.

on

## TERMMSG

---

## Chapter 3. Monitoring Routines

SA z/OS offers several routines that can be used to monitor various aspects of your enterprise.

---

### AOFADMON

#### Purpose

The AOFADMON routine is used to determine the status of a job within the operating system using the MVS D A method.

It is strongly recommended that you use INGPJMON rather than AOFADMON.

#### Syntax

▶▶—AOFADMON—*jobname*—————▶▶

#### Parameters

*jobname*

The job name that the operating system knows the associated application by.

#### Return Codes

- 0 The job is active.
- 4 The job is starting.
- 8 The job is inactive.
- 12 Parameter error.

---

### AOFAPMON

#### Purpose

The AOFAPMON routine is used to determine the status of a PPI receiver. It calls DISPPI and checks if a specific PPI receiver is active.

#### Syntax

▶▶—AOFAPMON—*pname*—————▶▶

#### Parameters

*pname*

The name of the PPI receiver this routine searches for. When the PPI receiver is active, the system issues return code 0. Otherwise return code 8 is issued.

#### Restrictions and Limitations

None.

## AOFAPMON

### Return Codes

- 0 The resource is active.
- 8 The resource is inactive.

---

## AOFATMON

### Purpose

The AOFATMON routine is used to determine the status of a task operating within the NetView environment. When the application is defined using the SA z/OS customization dialogs, the application job name must be defined to be the NetView task name.

### Syntax

▶▶—AOFATMON—*taskname*—————▶▶

### Parameters

*taskname*

The name of the NetView task whose status is to be obtained. This name is the same as the application job name.

### Return Codes

- 0 The task is active.
- 4 The task is starting.
- 8 The task is inactive.
- 12 Parameter error.

---

## AOFCPSM

### Purpose

The AOFCPSM routine is used to determine the status of processor operations.

### Syntax

▶▶—AOFCPSM—*jobname*—————▶▶

### Parameters

*jobname*

The job name that SA z/OS knows the processor operations application by.

### Return Codes

- 0 The task is active.
- 8 The task is inactive.
- 12 Error.



---

## AOFNCOMON

### Purpose

The AOFNCOMON routine is used to determine the status of the NETCONV connection running between the NMC server and z/OS NetView. The connection type can either be a TCPIP or SNA connection. It runs on the related work operator taking care of it.

### Syntax

▶▶—AOFNCOMON—*jobname*—————▶▶

### Parameters

*jobname*

The job name that automation knows the associated application as. This can be obtained from the SUBSJOB task global variable that is returned by AOCQRY.

### Return Codes

- 0 The connection is active.
- 8 The corresponding work operator does not hold a connection. The connection is inactive.
- 12 The connection status cannot be determined.

---

## AOFUXMON

### Purpose

The AOFUXMON routine is used to determine the status of a resource with application type USS. This resource can either be a z/OS UNIX process, a file in the z/OS UNIX file system (HFS), or a TCP port. Depending on the kind of resource (process, file, or port) AOFUXMON decides which internal monitoring method to use.

### Syntax

▶▶—AOFUXMON—*jobname*—————▶▶

### Parameters

*jobname*

The job name that SA z/OS knows the associated USS process, file, or port by.

### Restrictions and Limitations

AOFUXMON should only be used as a programming facility because its only output is a return code.

AOFUXMON uses active rather than passive monitoring for ports. Active monitoring will cause a connection to be established to an active port. If this is not desirable then a customer supplied monitoring routine should be used instead of AOFUXMON for port monitoring.

## Return Codes

- 0 The resource is active.
- 4 The resource is starting.
- 8 The resource is inactive or OMVS is inactive.
- 12 One of the following parameter errors occurred:
  - The *jobname* parameter was not specified.
  - The *jobname* parameter does not represent a USS type resource.
  - The *jobname* parameter does not represent a USS PATH, PORT or FILE.
- 20 A return code other than 0, 4 or 8 was returned from the USS INGCCMD routine. Check for related messages or turn on debug for AOFUXMON (this also turns on debug for INGCCMD).

---

## INGPJMON

### Purpose

The INGPJMON routine is used to determine the status of a job as known by the operating system. This is *not* the SA z/OS status of the job, which should be determined using AOCQRY.

INGPJMON does the following:

- It optionally returns the jobname and address space ID that match passed criteria
- It allows you to search for all address spaces that match the specified jobname
- It supports optional address-space search criteria

This monitoring routine is the foundation for supporting duplicate job names because standard address space monitoring takes the address space ID associated with the job into account. This allows you to distinguish between multiple occurrences of the same job in the system.

### Syntax

```

▶▶—INGPJMON—jobname— [,asid] [,stem] [,options]—▶▶

```

**Note:** All parameters are positional and must be replaced by a comma if omitted and followed by another operand.

### Parameters

#### *jobname*

This is the name of the job to be searched for. An asterisk (\*) must be specified as a placeholder if no job name is given.

#### *asid*

This is the address space ID (in hex) associated with the job. If omitted, the INGPJMON routine returns the first address space that matches the job name.

#### *stem*

This is the name of a NetView task global stem variable that will contain the job name and ASID of the address space that has been found.

The parameter is optional. If a task global name is specified, the following data are returned separated by a comma:

1. Job name.
2. Address space ID. If more than one ASID are returned, they are separated by a blank.

#### *options*

These are additional options, as follows:

#### **\*ALL**

Causes the monitoring routine to return all ASIDs that match the specified job name.

#### **\*TRACE**

Causes the monitoring routine to trace its processing by means of the component trace.

The following example shows how to retrieve data via a NetView task global stem:

```
'INGPJMON WEBSERVER,,STEM01,*ALL'
'GLOBALV GETT STEM01.0 STEM01.1'
exit 0
```

After executing the sample REXX the following data will be returned:

```
stem01.0 = 1
stem01.1 = WEBSERVER,0028 0033 0045,
```

In this example WEBSERVER is running in three address spaces.

A maximum of 48 ASIDs can be returned in each stem variable due to the NetView restriction of 256 bytes per variable. If more than 48 ASIDs are returned then additional ASIDs will be returned in additional stem variables. The last ASID is terminated by a comma.

## Return Codes

- 0 The job is active.
- 4 The job is starting.
- 8 The job is inactive.
- 12 Parameter error.

---

## INGPSMON

### Purpose

The INGPSMON routine is used to determine the status of an MVS subsystem. Unlike INGPJMON it does not search MVS address space control blocks but monitors the status of the specified job name via the IEFSSI service.

### Syntax

```
►►—INGPSMON—jobname— [,varname] [,options]—►
```

## Parameters

### *jobname*

This is the job name that is assigned to the subsystem. The job name must be identical to the MVS subsystem name. Specify !PRI for the primary subsystem.

### *varname*

This is the name of a NetView task global stem variable that contains information about the MVS subsystem. The output that is returned in the task global variable is as follows:

| Byte | Length | Description                                    |
|------|--------|------------------------------------------------|
| 1    | 4      | Name of MVS subsystem                          |
| 5    | 1      | Delimiter, contains blank                      |
| 6    | 1      | Contains P if primary subsystem                |
| 7    | 1      | Contains D if dynamic                          |
| 8    | 1      | Contains S if subsystem accepts SETSSI command |
| 9    | 1      | Contains A if subsystem is active              |

### *options*

These are additional options, as follows:

#### **\*TRACE**

Causes the monitoring routine to trace its processing by means of the MVS component trace function.

## Return Codes

- 0** The job is active.
- 4** The job is starting.
- 8** The job is inactive.
- 12** Parameter error.

---

## INGROMON

### Purpose

The INGROMON routine is used to determine the status of the OMVS address space.

### Syntax

▶▶—INGROMON—▶▶

### Return Codes

- 0** OMVS is fully initialized.
- 4** OMVS is starting. INGROMON will always ensure that the AT is loaded prior to returning this code.
- 8** OMVS is inactive.

---

## INGVMON

### Purpose

The INGVMON routine is used to determine the status of a virtual operator station task (VOST). It should be used as monitoring routine in a VOST management APL.

### Syntax

►►—INGVMON—*jobname*—◄◄

### Parameters

*jobname*

Specifies the job name that SA z/OS knows the associated VOST management APL as. This can be obtained from the SUBSJOB task global variable returned by AOCQRY.

### Restrictions and Limitations

INGVMON should only be used as a programming facility because its only output is a return code.

### Return Codes

- 0 The VOST is ACTIVE.
- 8 The VOST is INACTIVE.
- 12 Monitoring failed.
- 16 One of the following parameter errors occurred:
  - The *jobname* parameter was not specified.
  - The *jobname* parameter is not a valid job name.

---

## ISQMTSYS

### Purpose

The ISQMTSYS routine monitors processor operations target system resources. It is used to verify the availability of a target system according to a timer defined by the user.

### Syntax

►►—ISQMTSYS—*jobname*—◄◄

### Parameters

*jobname*

The job name that SA z/OS knows the processor operations target system by.

### Return Codes

- 0 The target system is active.
- 4 The target system is starting.
- 8 The target system is inactive.
- 12 The resource could not be found.



---

## Chapter 4. ING\$QRY NetView Automation Table Function

---

### ING\$QRY

#### Purpose

SA z/OS provides a NetView Automation Table Function (ATF), called ING\$QRY. This allows you to query or compare the status and other important attributes of jobs that are controlled by SA z/OS from within the AT and use the result as a condition in the AT statement. INGQUERY is an alias of ING\$QRY.

Refer to “the attribute parameter description” for the attributes that can be queried and used in an AT entry. The routine returns the attribute or comparison result as the function value of the NetView ATF function so that it can be used within the AT statement.

#### Format

►► ATF—(' ING\$QRY—*attribute*—*comp\_item*—*JOB=jobname*—*APPL=subsys*—')=value—◄◄

#### Parameters

The program name and the parameters must be specified with a literal quoted string. However, variable values can be passed as ATF parameters using the VALUE (varname) syntax.

##### *attribute*

This identifies the particular attribute of the job. It can be one of the following:

##### **APPL**

The subsystem name of the resource. This assumes that the specified resource name is a job name. It can be used to check if the job is controlled by SA z/OS.

##### **ASID**

The address space ID of the resource.

##### **CATEGORY**

The category of the resource (for example, CICS, IMS, DB2, and so on).

##### **CMDPFX**

The command prefix of the resource.

##### **FILE**

The file information of the resource.

##### **FILTER**

Returns information about the command parameters that are specified to make the process unique.

##### **IPSTACK**

Returns the IP stack name of the resource.

**JOB**

The name of the job. It can be used to check whether the resource (subsystem) is managed by SA z/OS.

**JOBTYPE**

The job type of the resource (MVS, NONMVS or TRANSIENT).

**OPER**

The work operator that is associated with the resource.

**OWNER**

The owner information of the resource.

**PARENT**

The parent information for the resource. Parent information is derived from the HasParent relationship that has a sequence number assigned to it.

**PATH**

Information about the UNIX process that the resource represents.

**PID**

The USS Process ID (PID) that is associated with the resource.

**PLEX**

Returns the name of the plex that is associated with the resource.

Currently, this attribute is only used for the IMSplex name as specified in the IMS CONTROL Policy Item.

**PORT**

The TCPIP port that is associated with the resource.

**PROCESS**

Contains START or STOP if the resource is within the startup or shutdown phase, respectively.

**STAT**

The agent status of the resource.

**SUBCAT**

The subsystem subcategory (for example, IRLM, TRACKER, TOR, AOR, and so forth).

**SUBID**

The MVS subsystem identifier of the resource.

**SYMBOL<sub>*n*</sub>**

Returns the requested application symbol, where *n* is 1–9.

**USER**

The USS user ID that is associated with the resource.

**WLMNAME**

The WLM resource name that is associated with the resource.

**WTOR**

This attribute returns all outstanding reply IDs.

***comp\_item***

Defines the string that the attribute value should be compared against. More than one compare item can be specified, separated by a blank character. The compare item can be a wildcard, for example, abc\*

If the attribute value matches one of the items in the compare string, the ATF value is set to string 'TRUE', otherwise it is set to string 'FALSE'.



If no compare item is specified, the ING\$QRY ATF function returns the attribute value.

*jobname*

The name of the job that the attribute value should be returned for or compared against. The default is the job that issued the message.

*subsys*

The name of the subsystem that the attribute value should be returned for or compared against.

## Return codes

The ING\$QRY routine sets the following return code when returning to NetView:

- 0 Normal completion.
- 1 The specified variable name is unknown.
- 2 An error occurred and no output was provided. This may be due to:
  - Invalid parameters were passed to the routine.
  - Returning the common global variable failed.
  - The job does not belong to a subsystem that is controlled by SA z/OS.

## Restrictions

None.

## Usage

This routine is used as a NetView Automation Table Function (ATF) within an AT condition statement to query and compare attributes of SA z/OS-controlled subsystems.

## Examples

1. The following example checks whether the status of the resource that issued message WAS001I is UP and if so triggers exit AOFRIMSG:

```
IF MSGID = 'WAS001I' & ATF('ING$QRY STAT') = 'UP' THEN
EXEC(CMD('AOFRIMSG') ROUTE(ONE *));
```

2. The following example returns the status of the resource that issued message WAS002I in variable MYVAR:

```
IF MSGID = 'WAS002I' & ATF('ING$QRY STAT') = MYVAR THEN
EXEC(CMD('AOFRIMSG 'MYVAR) ROUTE(ONE *));
```

3. The following example checks whether the status of the resource that issued message WAS003I is either DOWN or UP. If so, it triggers exec AOFRIMSG:

```
IF MSGID = 'WAS003I' & ATF('ING$QRY STAT,DOWN UP') = 'TRUE' THEN
EXEC(CMD('AOFRIMSG') ROUTE(ONE *));
```

4. The following example checks whether the status of job AMY0 is UP when message WAS004I is issued and if so, issues command INGLIST:

```
IF MSGID = 'WAS004I' & ATF('ING$QRY STAT,UP,JOB=AMY0') = 'TRUE' THEN
EXEC(CMD('INGLIST AM* OUTMODE=NETLOG') ROUTE(ONE *));
```

5. The following example uses a wildcard in the compare items list. In this case it would return a list of AUTO... matches:

```
IF MSGID = 'WAS005I' & ATF('ING$QRY STAT,AU* DOWN') = 'TRUE' THEN
EXEC(CMD('RES') ROUTE(ONE *));
```

6. The following example checks whether the subsystem is controlled by SA z/OS, assuming that the 2nd token of the message contains the job name:

```
IF MSGID = 'WAS006I' & TOKEN(2) = SVJOB & ATF('ING$QRY JOB') = VALUE(SVJOB) THEN
EXEC(CMD('AOFRIMSG') ROUTE(ONE *));
```



---

## Part 3. SA z/OS I/O Operations Commands

|                                                            |     |                                                                            |     |
|------------------------------------------------------------|-----|----------------------------------------------------------------------------|-----|
| <b>Chapter 5. I/O Operations Commands (API)</b> . . . . .  | 193 | QUERY RELATION DEV . . . . .                                               | 244 |
| Using I/O Operations Commands for<br>Programming . . . . . | 193 | QUERY RELATION HOST . . . . .                                              | 246 |
| Calling the I/O Operations API . . . . .                   | 193 | QUERY RELATION SWITCH . . . . .                                            | 246 |
| Safe Switching . . . . .                                   | 195 | QUERY SWITCH . . . . .                                                     | 247 |
| FICON Switches . . . . .                                   | 195 | REMOVE and RESTORE CHP . . . . .                                           | 250 |
| FICON Cascaded Switches . . . . .                          | 195 | REMOVE DEV and RESTORE DEV . . . . .                                       | 254 |
| Common Elements . . . . .                                  | 196 | WRITEFILE . . . . .                                                        | 259 |
| Common Syntax Elements . . . . .                           | 196 | WRITEPORT . . . . .                                                        | 261 |
| Common Parameters . . . . .                                | 197 | WRITESWCH . . . . .                                                        | 266 |
| Common Query Commands Syntax . . . . .                     | 201 |                                                                            |     |
| Common Query Entity/Interface Output<br>Header . . . . .   | 203 | <b>Chapter 6. Invoking I/O Operations using the<br/>API</b> . . . . .      | 273 |
| Common Query Relation Output Format . . . . .              | 204 | API Calls by REXX EXECs . . . . .                                          | 273 |
| DELETE FILE . . . . .                                      | 209 | Rules for Calls by a REXX EXEC . . . . .                                   | 273 |
| QUERY ENTITY CHP . . . . .                                 | 210 | Two Examples of REXX EXEC Calls . . . . .                                  | 274 |
| QUERY ENTITY CNTLUNIT . . . . .                            | 215 | Generalized Example of a REXX EXEC Call . . . . .                          | 274 |
| QUERY ENTITY DEV . . . . .                                 | 218 | API Calls by the CALL Macro . . . . .                                      | 275 |
| QUERY ENTITY HOST . . . . .                                | 222 | General Information . . . . .                                              | 275 |
| QUERY ENTITY SWITCH . . . . .                              | 225 | The Parameter Lists . . . . .                                              | 275 |
| QUERY FILE . . . . .                                       | 228 | The Caller Should Check Register 15 Upon<br>Return From the Call . . . . . | 275 |
| QUERY INTERFACE CNTLUNIT . . . . .                         | 229 | Calling Program Uses IHVAPI2 . . . . .                                     | 276 |
| QUERY INTERFACE SWITCH . . . . .                           | 235 | Calling Program Uses IHVAPI . . . . .                                      | 278 |
| QUERY RELATION CHP . . . . .                               | 243 |                                                                            |     |
| QUERY RELATION CNTLUNIT . . . . .                          | 244 |                                                                            |     |

This part describes SA z/OS I/O operations (I/O-Ops) commands that are available through the API only.

For general information about the SA z/OS commands, see *IBM Tivoli System Automation for z/OS User's Guide*.

All commands described in *IBM Tivoli System Automation for z/OS Defining Automation Policy* are also available through the API.



---

## Chapter 5. I/O Operations Commands (API)

---

### Using I/O Operations Commands for Programming

In addition to the I/O-Ops commands described in *IBM Tivoli System Automation for z/OS Operator's Commands*, the following commands are available to programmed API calls:

- DELete File
- Query Entity Chp
- Query Entity CntlUnit
- Query Entity Dev
- Query Entity Host
- Query Entity Switch
- Query File
- Query Interface CntlUnit
- Query Interface Switch
- Query Relation Chp
- Query Relation CntlUnit
- Query Relation Dev
- Query Relation Host
- Query Relation Switch
- Query Switch
- Remove and Restore Chp
- Remove and Restore Dev
- WRITEFILE
- WRITEPORT
- WRITESWCH

### Calling the I/O Operations API

I/O-Ops application program interfaces support:

- Invocations from an EXEC written in the REXX programming language.
- Invocations from a user program written in a language that adheres to the Assembler Language CALL macro interface conventions used by MVS/ESA. This type of caller is referred to as a program that uses the CALL macro. For information on the CALL macro, refer to *MVS/ESA Application Development Macro Reference*. These callers can invoke IHVAPI; however IHVAPI2 is recommended.
- All variables, except arrays, data blocks, tables and tokens, must be in uppercase.
- Programs that use the CALL macro to invoke IHVAPI2 (preferred for the following reasons):
  - IHVAPI2 lets the caller choose between managing the command response area or letting I/O-Ops do so. IHVAPI requires the user to manage the response area.
  - IHVAPI2 can return data in a response area that exceeds 64KB; IHVAPI cannot.

## Using I/O Operations Commands for Programming

- IHVAPI2 accepts all the variables needed by the I/O-Ops commands, including multisystem commands. IHVAPI accepts only 24-character variables as input parameters except those that specify an array, data block, or table. For those operands, it accepts a variable long enough to contain the array or table.
- Tokens
- The MVS REXX Call invocation in addition to the Address Link invocation.
- TSO/E (optional). For further information about how to invoke I/O-Ops by a REXX EXEC call, refer to Chapter 6, “Invoking I/O Operations using the API,” on page 273.

### General Information About the Response Area

For most commands, I/O-Ops returns data to the caller in a response area.

**The Data In the Response Area:** When data is returned in the response area, it is either a single record or a concatenation of records in character or hexadecimal format, or both, which overlays any previous data.

For most commands, I/O-Ops returns at least one message in the response area. However, there are exceptions. For example, the multisystem commands can return no data, one or more messages, or a data block. Also, failed commands do not always return data in the response area.

When I/O-Ops returns a message, the first 3 characters are IHV, which identify I/O-Ops. Although the messages resulting from most commands are concatenated, up to four blank 80-character records can intervene between two successive I/O-Ops messages.

**The Length of the Response Area:** The amount of data that can be returned by a multisystem command can be very large, so the following approximate maximum lengths are provided.

For DISPLAY DEVICE, DISPLAY RESULTS, and DISPLAY VARY commands, assume that 65,528 bytes (64KB) suffice.

For REMOVE DEV, RESTORE DEV, and the QUERY commands, calculate  $100 + (1 + x) * y * z$ , where:

- x** Is one of the following:
  - The number of objects in a QUERY ENTITY, REMOVE DEV, or RESTORE DEV command
  - The number of interfaces in a QUERY INTERFACE command
  - The number of paths in a QUERY RELATION command (in this context, number is the number from one host's perspective)
- y** The number of hosts scoped in the command
- z** The size of the output row (the following sizes are approximations):
  - 250 for a REMOVE DEV or RESTORE DEV command
  - 300 for a QUERY ENTITY or QUERY INTERFACE command
  - 500 for a QUERY RELATION command

For all other commands, assume that 25,600 bytes (24KB) suffice.

For invocations by a REXX EXEC, the final size should be doubled because I/O-Ops uses the IRXEXCOM facility to access ihvrc, ihvreas, and ihvresp, and it uses the STORE function of IRXEXCOM to set them.

---

## Safe Switching

I/O-Ops varies paths online or offline when, because of port manipulation, the path from a channel to a device either becomes valid or is no longer valid.

The term *safe-switching* means that *all* vary path offline requests due to an I/O-Ops connectivity command are backed out if *one* of these requests fails and BACKOUT was specified at command invocation. All requests means those requests on all systems that have access to the switch (or switches) that are affected by the command.

For FICON® switches, safe-switching also includes the entire vary process for connectivity commands that affect Inter-Switch-Link ports (E-ports). Because I/O-Ops does not know the topology between the entry switch and the destination switch of a path, paths that go through an ISL link will not be varied when an E-port is the target of a connectivity command.

The following conditions result in the failure of a request:

- A vary path offline request fails when the request would disable the last path to a device that is currently in use.
- If no VTAM connection could be established between two systems that have access to a switch and run I/O-Ops, I/O-Ops on the local system (that is, where the command is entered) assumes that the command fails on the remote system. To avoid this, exclude this system from consensus processing using the command RESET HOST *vtamname* PURGE.
- For other reasons refer to the section “Making Connectivity Changes” in the appendix, “Definitions for I/O Operations Commands” in *IBM Tivoli System Automation for z/OS Operator’s Commands*.

---

## FICON Switches

FICON switches allow imbedded space characters on port names. Consequently, I/O-Ops will not issue message IHVD106I when detecting imbedded blanks in port names of FICON switches.

However, I/O-Ops does not support imbedded blanks on port names, either in the ISPF dialogs or in the console command interface. The reason is that generic names and port names must not contain imbedded blanks when used in I/O-Ops console commands.

---

## FICON Cascaded Switches

I/O-Ops supports cascaded switches with some restrictions:

1. For CTC connections on cascaded switches, I/O-Ops can neither display CTC control unit data nor manage CTC devices. The reason for this is that when I/O-Ops attempts to determine the attached NDs of such a device, it can get stuck behind a never-ending channel program on the device.
2. The Block command is not supported on Inter-Switch-Link ports (E\_Ports). When an E\_Port is affected by the command, it is rejected with return code 8 and reason code X'49'. In addition, the message IHVC913I is issued, showing the first or only port that is affected by the command.
3. All other I/O-Ops commands affecting E\_Ports (Allow, Prohibit, Unblock, and WRITEPORT) must specify the command option IGNore when an E\_Port is involved. Otherwise the command is rejected with return code 8 and reason

## Using I/O Operations Commands for Programming

code X'49'. In addition, the message IHVC913I is issued, showing the first or only port that is affected by the command.

The IGNore option makes the issuer of the command aware that *safe-switching* can no longer be guaranteed.

4. If an attached Node Descriptor of a device cannot be determined because the path or channel is offline, the Display Device command does not show any control unit data for the particular channel path id.
5. A dynamic configuration change that results in the allocation or deallocation of a cascaded switch is currently *not* supported.

**Note:** It is recommended that all switches are defined to the Hardware Configuration Definition (HCD) including their device numbers. This allows I/O-Ops to also show the LSN for cascaded switches.

---

## Common Elements

Many of the commands described in this chapter share several elements (such as syntax, parameters, output headers, etc.) that are described in this section. To help make the command descriptions that follow a little clearer, the descriptions of these common parameters will not be repeated. Instead, you will be referred to the relevant part of this section.

The common elements are:

- Various syntax elements, see “Common Syntax Elements”
- Various parameters, see “Common Parameters” on page 197
- The syntax for Query Entity/Interface/Relation commands, see “Common Query Commands Syntax” on page 201
- Output header for all Query Entity/Interface output structures, see “Common Query Entity/Interface Output Header” on page 203
- Output for all Query Relation commands, see “Common Query Relation Output Format” on page 204

## Common Syntax Elements

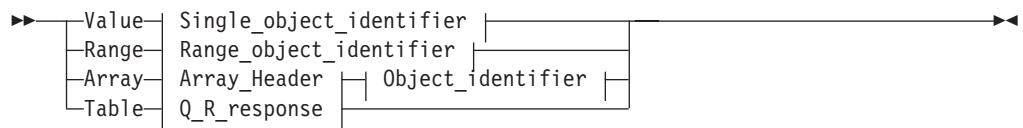
The following syntax elements are common to several commands:

- Object Format
- Single\_object\_identifier
- Range\_object\_identifier
- Scope
- Host\_object\_identifier

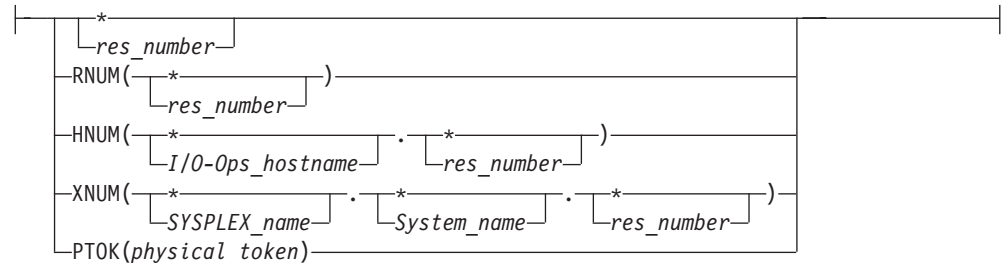
In the syntax diagrams for the commands in this chapter these syntax elements are shown only as syntax fragments.



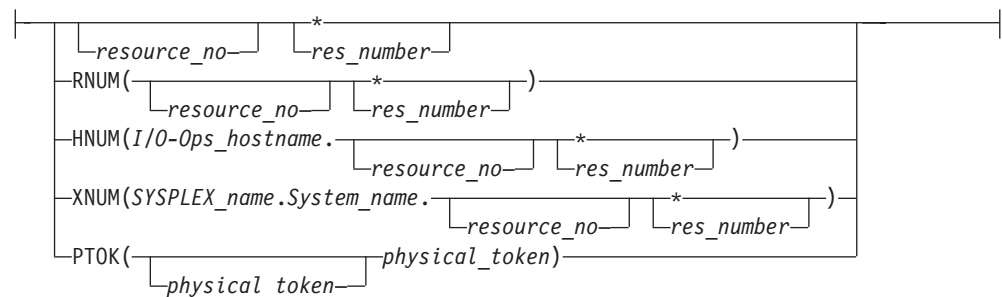
**Object Format**



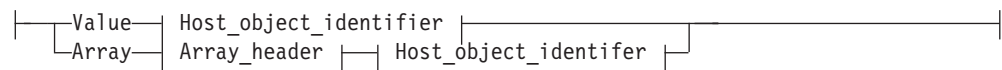
**Single\_object\_identifier:**



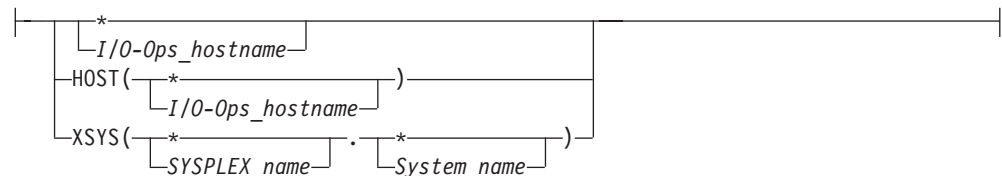
**Range\_object\_identifier:**



**Scope:**



**Host\_object\_identifier:**



**Common Parameters**

These common parameters are:

- Object Identifier (see “Object Identifiers” on page 198):
  - Value
  - Range
  - Array
  - Table

## Common Elements

- I/O resource identifiers (see “I/O Resource Identifiers”):
  - RNUM
  - HNUM
  - XNUM
  - PTOK
  - LTOK
- Host identifiers (see “Host Identifiers” on page 200):
  - HOST
  - XSYS
  - SCOPE

### Object Identifiers

**Value** *value* | \*

Specify V or VALUE. Then, specify either a single *element* or asterisk (\*) for all *elements* known to the issuing I/O-Ops. (If \* is specified, output array elements are sorted by searched element.)

**Note:** An element can either be a CHIPID, CONTROL UNIT or a DEVICE NUMBER.

**Range** *lower-upper* | *lower* - \*

Specify R or RANGE. Specify the lower limit, followed by a hyphen, followed either by the upper limit of the range or an asterisk '\*' to specify the highest number. Output array elements are sorted by number.

**Array**

Specify A or ARRAY. The output of array elements is returned in the input order. Specify the input array in the following format:

Table 9. Standard SA z/OS Array Format

| Offset         |     |           |     |           |                                                                                    |
|----------------|-----|-----------|-----|-----------|------------------------------------------------------------------------------------|
| Dec            | Hex | Type      | Len | Name(Dim) | Description                                                                        |
| 0              | (0) | UNSIGNED  | 4   | NUM_ROWS  | Number of elements                                                                 |
|                |     | 1... ..   |     | FMT       | Array format                                                                       |
| 4              | (4) | UNSIGNED  | 1   | FMT_ID    | 1 SA z/OS formatted array<br>Format identifier. Only 0 is valid for SA z/OS arrays |
| 5              | (5) | CHARACTER | 3   | *         | Reserved                                                                           |
| Array elements |     |           |     |           |                                                                                    |
| 8              | (8) | CHARACTER | 38  | OBJ_ID(*) | Object identifier                                                                  |

**Notes:**

1. Up to 32767 (decimal) CHPIDs can be entered for CHP, but for CNTLUNIT and DEV the overall size is restricted to 32000 control unit numbers or device numbers respectively.
2. SA z/OS continues to support the ESCON® Manager Release 2 format for input arrays. For information about this format, see *Using the Enterprise Systems Connection Manager*.

**Table**

Specify T or TABLE. Requires CODE=1 and the RESPONDER host application name must match the scope host name to be operated on. The table format is identical to the output format of a QUERY RELATION command.

### I/O Resource Identifiers

An I/O resource identifier type can be one of the following:

**RNUM**

RNUM is an identifier value that is a resource number. All I/O resource object types have a resource number. Each object type specifies it differently, such as:

**CHP**

Channel path identifier

**CU** Control unit number

**DEV**

Device number

**SWITCH**

Switch device number

The value for RNUM must be from 1 to 4 hexadecimal characters or an asterisk (\*).

Examples:

```
*
40
40-*
40-4F
RNUM(*)
RNUM(100)
RNUM(100-*)
RNUM(100-10F)
```

**HNUM**

HNUM is an identifier value that is qualified by an I/O-Ops host name. It is a specific host's resource number. HNUM must be 1 to 8 alphanumeric characters for the host name (or be an asterisk), followed by a period (.), followed by 1 or 2 hexadecimal characters for the resource number (or be an asterisk).

Examples:

```
HNUM(*.*)
HNUM(*.40)
HNUM(H1.*)
HNUM(H1.40)
HNUM(H1.40-*)
HNUM(H1.40-4F)
```

**XNUM**

XNUM is an identifier value that is qualified by a sysplex name and a system name. It is a specific sysplex system's resource number. XNUM must be 1 to 8 alphanumeric characters for the sysplex name (or be an asterisk), followed by a period (.), followed by 1 to 8 alphanumeric characters for the system name (or be an asterisk), followed by a period (.), followed by 1 or 2 hexadecimal characters for the resource number (or be an asterisk).

Examples:

```
XNUM(*.*.*)
XNUM(*.S1.40)
XNUM(X1.*.40)
XNUM(X1.S1.*)
XNUM(X1.S1.40)
XNUM(X1.S1.40-*)
XNUM(X1.S1.40-4F)
```

**PTOK**

PTOK is an identifier value that is a physical token. PTOK is a 32-character field. Refer to *IBM Tivoli System Automation for z/OS Operator's Commands* for further information about physical tokens.

Examples:

## Common Elements

```
PTOK(... 009032002IBM020000000000100)
PTOK(... 009032002IBM020000000000100-...009032002IBM0200000999999900)
```

### L TOK

L TOK is an identifier value that is a logical token. L TOK is a 32-character field. Refer to *IBM Tivoli System Automation for z/OS Operator's Commands* for further information about logical tokens.

Examples:

```
L TOK(0123456789ABCDEF0123456789ABCDEF)
L TOK(0123456789ABCDEF0123456789ABCDEF-0123456789ABCDEF0123456789ABCDEF)
```

## Host Identifiers

A host identifier type can be one of the following:

### HOST

HOST is an identifier value that is the VTAM application name or TCP/IP host name of an I/O-Ops. HOST must be 1–8 alphanumeric characters for the host name (or be an asterisk).

Examples:

```
HOST(*)
HOST(H1)
HOST(H1-*)
HOST(H1-H9)
```

### XSYS

XSYS is an identifier value that is a sysplex name, or a system name, or both. The rules are:

- If both a sysplex name and system name are specified, only that system in the sysplex is considered for the command
- If a specific sysplex name is specified, only the systems in that sysplex are considered for the command
- If a specific system name is specified, only the systems with that name are considered for the command

XSYS must be from 1 to 8 alphanumeric characters for the sysplex name (or be an asterisk), followed by a period (.), followed by 1 to 8 alphanumeric characters for the system name (or be an asterisk).

Examples:

```
XSYS(*.*)
XSYS(*.S1)
XSYS(X1.*)
XSYS(X1.S1)
XSYS(*.S1-*)
XSYS(*.S1-S9)
XSYS(X1.S1-*)
XSYS(X1.S1-S9)
```

### SCOPE

SCOPE specifies the set of I/O-Ops hosts that respond to a multisystem command.

### NOPATHTEST

No checking is done on the command to verify that the path from the CHPID to the device exists.

**Note:** For QUERY RELATION HOST, the NOPATHTEST option is only valid on the QUERY RELATION HOST to Device command.

**PATHTEST**

If you specify Pathtest on the command, checking is done to verify that the device is physically there with relation to each CHPID.

**Note:** For QUERY RELATION HOST, the PATHTEST option is only valid on the QUERY RELATION HOST to Device command.

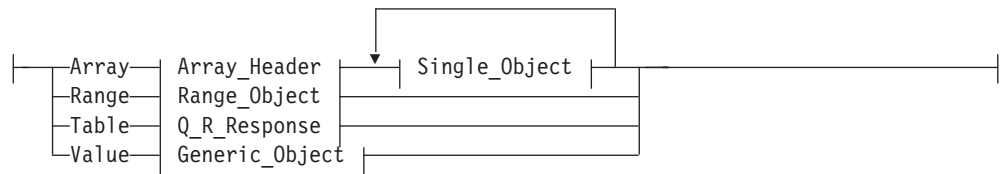
**Notes:**

1. When ARRAY is the *Object\_format\_type*, the *Object\_Identifier\_Types* can be mixed and every *Object\_Identifier\_Type* must match the class of the specified *Object\_Type* (all must be I/O\_resources or all must be Hosts). For example, Q E HOST can accept only HOST and XSYS entries in the array.
2. The Array\_header contains the number of elements in the array.
3. PTOK is valid with RANGE but you should be fully aware of PTOK structure. For example, RANGE PTOK could be used to specify all of the serial numbers of a certain type of device. However, certain PTOK values may cause unpredictable results with RANGE.
4. When ARRAY is the *SCOPE\_format\_type*, the *Host\_Object\_Identifier\_Types* can be mixed (HOST and XSYS).
5. Output from a QUERY ENTITY command consists of a header, which is identical for each entity with the exception of the "Eye-Catcher" (offset 0), followed by the substructures, which are unique to each type of entity.
6. If not otherwise stated at the particular command descriptions no input port information is returned by a QUERY RELATION command when the command specifies a switch that is the destination switch of a cascaded switch pair.

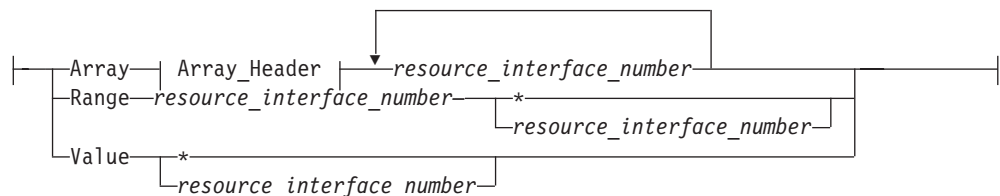
**Common Query Commands Syntax**

The following syntax is common for all Query Entity/Interface/Relation commands.

**(Host\_)Entity\_Object:**

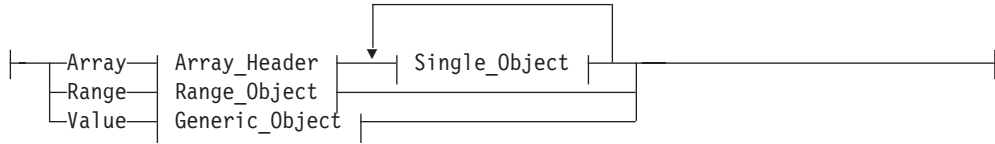


**Interface\_Object:**

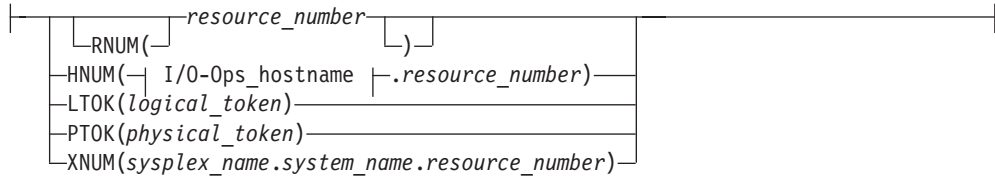


**(Host\_)Relation\_Object:**

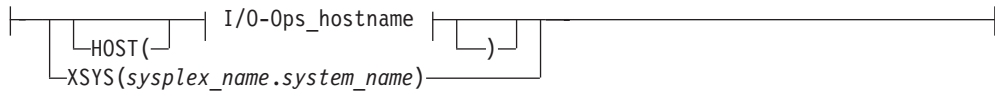
## Common Elements



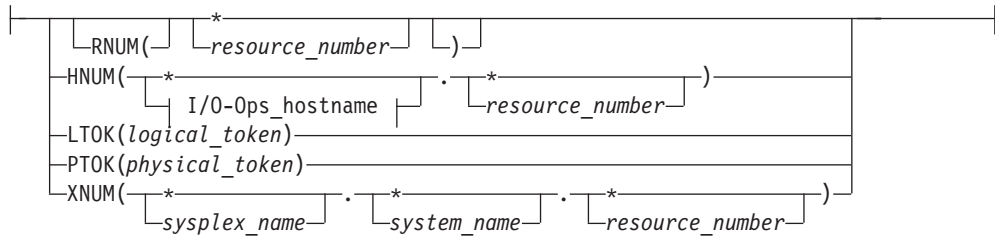
### Single\_Object (when object type is I/O resource):



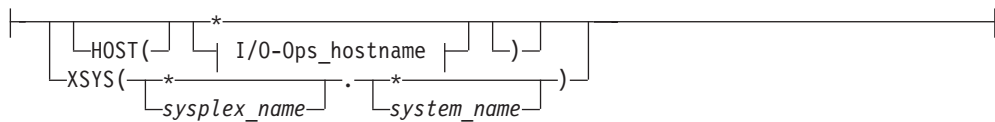
### Single\_Object (when object type is HOST):



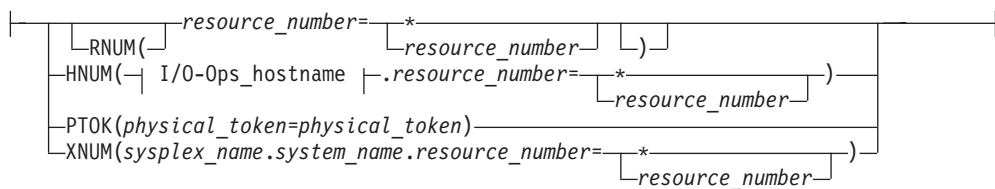
### Generic\_Object (when object type is I/O resource):



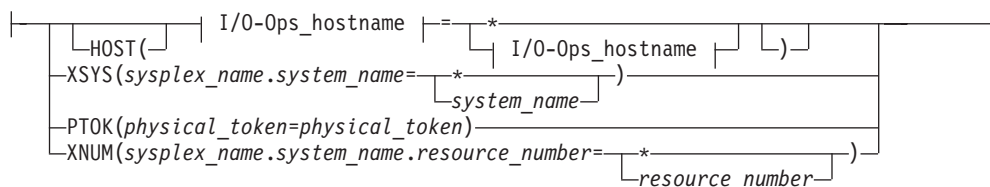
### Generic\_Object (when object type is HOST):



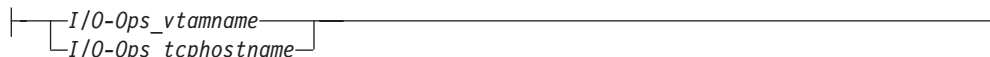
### Range\_Object (when object type is I/O resource):



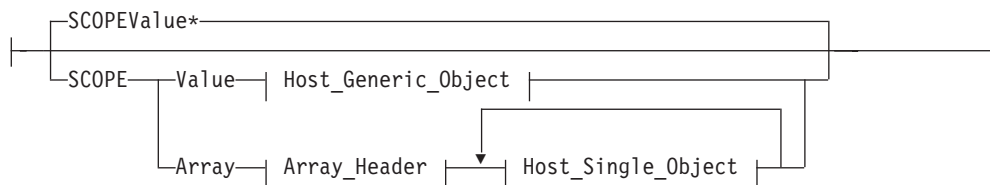
### Range\_Object (when object type is HOST):



**I/O-Ops\_hostname:**



**Scope:**



**Options:**



**Common Query Entity/Interface Output Header**

Table 10 shows the common output header that is produced for all QUERY ENTITY/INTERFACE output structures.

Table 10. Header for all Query Entity/Interface Output Formats

| Offset |      |           |     |             |                                                                                                                                                                                                                                 |
|--------|------|-----------|-----|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dec    | Hex  | Type      | Len | Name(Dim)   | Description                                                                                                                                                                                                                     |
| 0      | (0)  | STRUCTURE | 80  | HDR         |                                                                                                                                                                                                                                 |
| 0      | (0)  | CHARACTER | 4   | EYE_CATCHER | Identifies the control block:<br>QEC Query Entity Chp<br>QED Query Entity Device<br>QEH Query Entity Host<br>QES Query Entity Switch<br>QEU Query Entity Cntlunit<br>QIS Query Interface Switch<br>QIU Query Interface Cntlunit |
| 4      | (4)  | UNSIGNED  | 2   | HDR_SIZE    | Size of this header                                                                                                                                                                                                             |
| 6      | (6)  | UNSIGNED  | 2   | ROW_SIZE    | Size of array element                                                                                                                                                                                                           |
| 8      | (8)  | CHARACTER | 8   | ESCM_HOST   | Responding host VTAM application name or TCP/IP host name                                                                                                                                                                       |
| 16     | (10) | CHARACTER | 4   | ESCM_REL    | SA z/OS version and release                                                                                                                                                                                                     |
| 20     | (14) | CHARACTER | 32  | HOST_PID    | Host physical identifier                                                                                                                                                                                                        |
| 20     | (14) | BITSTRING | 1   | *           |                                                                                                                                                                                                                                 |
|        |      | 111. .... |     | VALIDITY    | 0 = Valid<br>1 = Not current<br>2 = Not valid                                                                                                                                                                                   |
|        |      | ...1 1111 |     | *           | Reserved                                                                                                                                                                                                                        |

## Common Elements

Table 10. Header for all Query Entity/Interface Output Formats (continued)

| Offset |      |              |     |              |                                                                                                                                                |
|--------|------|--------------|-----|--------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Dec    | Hex  | Type         | Len | Name(Dim)    | Description                                                                                                                                    |
| 21     | (15) | CHARACTER    | 3   | *            | Reserved                                                                                                                                       |
| 24     | (18) | CHARACTER    | 6   | TYPE_NUM     | Processor type, for example, 002064                                                                                                            |
| 30     | (1E) | CHARACTER    | 3   | MODEL_NUM    | Processor model, for example, 108                                                                                                              |
| 33     | (21) | CHARACTER    | 3   | MFR          | Manufacturer, for example, IBM                                                                                                                 |
| 36     | (24) | CHARACTER    | 2   | PLANT        | Where manufactured                                                                                                                             |
| 38     | (26) | CHARACTER    | 12  | SEQUENCE_NUM | Serial number                                                                                                                                  |
| 50     | (32) | BITSTRING    | 2   | STATUS       | Status of PID                                                                                                                                  |
|        |      | 1... ..      |     | AMBIGUOUS    | Ambiguous state detected on PID                                                                                                                |
|        |      | .1... ..     |     | REFLECTED    | PID is derived from attached ND                                                                                                                |
|        |      | ..11 1111 >> |     | *            | Reserved                                                                                                                                       |
| 52     | (34) | UNSIGNED     | 4   | NUM_ROWS     | Dimension of array following this header                                                                                                       |
| 56     | (38) | UNSIGNED     | 1   | FORMAT_ID    | Identifies format of data                                                                                                                      |
| 57     | (39) | BITSTRING    | 1   | *            |                                                                                                                                                |
|        |      | 1... ..      |     | MORE_DATA    | 0 = All data that satisfies query is returned here.<br>1 = More data satisfies query (but won't fit now). Ask again with RANGE parameter type. |
|        |      | .1... ..     |     | TCP_HOST     | 1 = ESCM_HOST contains the TCP/IP host name                                                                                                    |
|        |      | .111 1111    |     | *            | Reserved                                                                                                                                       |
| 58     | (3A) | CHARACTER    | 8   | PLEX_NAME    | Sysplex name (blank if none)                                                                                                                   |
| 66     | (42) | CHARACTER    | 8   | SYST_NAME    | System name                                                                                                                                    |
| 74     | (4A) | CHARACTER    | 2   | *            | Reserved                                                                                                                                       |
| 76     | (4C) | UNSIGNED     | 4   | NUM_HOSTS    | Number of hosts responding                                                                                                                     |

## Common Query Relation Output Format

Table 11 shows the output that is common to all Query Relation commands.

Table 11. Output Format for all Query Relation Commands

| Offset |      |           |     |             |                                                                                                       |
|--------|------|-----------|-----|-------------|-------------------------------------------------------------------------------------------------------|
| Dec    | Hex  | Type      | Len | Name(Dim)   | Description                                                                                           |
| 0      | (0)  | STRUCTURE | *   | QRO         |                                                                                                       |
| 0      | (0)  | CHARACTER | 48  | HDR         | Header data                                                                                           |
| 0      | (0)  | CHARACTER | 4   | EYE_CATCHER | Identifies control block ('QRO')                                                                      |
| 4      | (4)  | UNSIGNED  | 2   | HDR_SIZE    | Length of (this) QRO.HDR                                                                              |
| 6      | (6)  | UNSIGNED  | 2   | ROW_SIZE    | Length of 1 ROW                                                                                       |
| 8      | (8)  | CHARACTER | 8   | ESCM_HOST   | Responding host VTAM application name or TCP/IP host name                                             |
| 16     | (10) | CHARACTER | 4   | ESCM_REL    | SA z/OS version and release                                                                           |
| 20     | (14) | UNSIGNED  | 4   | NUM_ROWS    | ROW dimension                                                                                         |
| 24     | (18) | UNSIGNED  | 1   | FORMAT_ID   | Identifies format of table                                                                            |
| 25     | (19) | BITSTRING | 1   | *           |                                                                                                       |
|        |      | 1... ..   |     | MORE_DATA   | 0 = Entire Query response in QRO<br>1 = Query response too large to fit in QRO (ask again, use RANGE) |
|        |      | .1... ..  |     | PATHTEST    | 1 = PATHTEST requested                                                                                |
|        |      | .1... ..  |     | TCP_HOST    | 1 = ESCM_HOST contains the TCP/IP host name                                                           |



Table 11. Output Format for all Query Relation Commands (continued)

| Offset                                                                                            |      |           |     |               |                                                                                                                                                                                                            |
|---------------------------------------------------------------------------------------------------|------|-----------|-----|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dec                                                                                               | Hex  | Type      | Len | Name(Dim)     | Description                                                                                                                                                                                                |
|                                                                                                   |      | ..11 1111 |     | *             | Reserved                                                                                                                                                                                                   |
| 26                                                                                                | (1A) | CHARACTER | 8   | PLEX_NAME     | Sysplex name (blank if none)                                                                                                                                                                               |
| 34                                                                                                | (22) | CHARACTER | 8   | SYST_NAME     | System name                                                                                                                                                                                                |
| 42                                                                                                | (2A) | CHARACTER | 2   | *             | Reserved                                                                                                                                                                                                   |
| 44                                                                                                | (2C) | UNSIGNED  | 4   | NUM_HOSTS     | Number of hosts responding                                                                                                                                                                                 |
| Path descriptions                                                                                 |      |           |     |               |                                                                                                                                                                                                            |
| 48                                                                                                | (30) | CHARACTER | 372 | ROW(*)        | Indexed by HDR.NUM_ROWS                                                                                                                                                                                    |
| 48                                                                                                | (30) | CHARACTER | 8   | HOST_APPL     | Host VTAM application name                                                                                                                                                                                 |
| 56                                                                                                | (38) | UNSIGNED  | 1   | CHPID         | Channel path identifier (00-FFx)                                                                                                                                                                           |
| 57                                                                                                | (39) | UNSIGNED  | 1   | PORTIN        | When data is flowing from the host, the input port on the switch (if a switch is in the path)                                                                                                              |
| 58                                                                                                | (3A) | UNSIGNED  | 2   | SW_DEVN       | Switch device number (if a switch is in the path).                                                                                                                                                         |
| 60                                                                                                | (3C) | UNSIGNED  | 1   | LSN           | Logical switch number (that goes with SW_DEVN) when a switch is in the path.                                                                                                                               |
| 61                                                                                                | (3D) | UNSIGNED  | 1   | PORTOUT       | When data is flowing from the host, the output port on the switch (if a switch is in the path)                                                                                                             |
| 62                                                                                                | (3E) | UNSIGNED  | 2   | CU_NUMBER     | Control unit number                                                                                                                                                                                        |
| 64                                                                                                | (40) | UNSIGNED  | 2   | DEV_NUMBER    | Device number                                                                                                                                                                                              |
| The following bits describe the validity of the data in the corresponding row                     |      |           |     |               |                                                                                                                                                                                                            |
| 66                                                                                                | (42) | BITSTRING | 2   | STATBITS      | Indicate row data validity                                                                                                                                                                                 |
|                                                                                                   |      | 1... ..   |     | VALID_DATA    | 1 = This row contains a valid path<br>0 = This row does not contain a valid path. Either the entity2 is not found in the database at all or there is no relation between the entity1 and entity2 specified |
|                                                                                                   |      | .1.. ..   |     | INCOMPLETE    | 0 = Queried data is in database (that is, not a proxy request)<br>1 = Queried data not known (that is, secondary host databases are not known)                                                             |
| The following bits describe switch data validity                                                  |      |           |     |               |                                                                                                                                                                                                            |
|                                                                                                   |      | ..1. .... |     | VALID_SW      | 1 = SW_DEVN is valid (switch either is or was operational)                                                                                                                                                 |
|                                                                                                   |      | ...1 .... |     | VALID_LSN     | 1 = LSN is valid (path is switched)                                                                                                                                                                        |
|                                                                                                   |      | .... 1... |     | VALID_PORTIN  | 1 = PORTIN value is verified                                                                                                                                                                               |
|                                                                                                   |      | .... .1.. |     | VALID_PORTOUT | 1 = PORTOUT value is verified                                                                                                                                                                              |
| The following bits indicate which path elements are detected to be involved in an ambiguous state |      |           |     |               |                                                                                                                                                                                                            |
|                                                                                                   |      | .... ..1. |     | AMBIG_PORTIN  | 1 = CHCH, CHCU detected on PORTIN                                                                                                                                                                          |
|                                                                                                   |      | .... ...1 |     | AMBIG_PORTOUT | 1 = CHCU detected on PORTOUT                                                                                                                                                                               |
| The following bits indicate whether ports (paths) are involved with chained or cascaded switches  |      |           |     |               |                                                                                                                                                                                                            |
| 67                                                                                                | (43) | 1... ..   |     | CHAIN_PORTIN  | 1 = PORTIN is part of CHAIN                                                                                                                                                                                |
|                                                                                                   |      | .1.. .... |     | CHAIN_PORTOUT | 1 = PORTOUT is part of CHAIN                                                                                                                                                                               |

## Common Elements

Table 11. Output Format for all Query Relation Commands (continued)

| Offset                                                                                                                                                                                                                                                                                                                                                                        |      |           |     |                |                                                                                                                                                                                              |  |  |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|-----------|-----|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|
| Dec                                                                                                                                                                                                                                                                                                                                                                           | Hex  | Type      | Len | Name(Dim)      | Description                                                                                                                                                                                  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                               |      | ..1. .... |     | VALID_DEVNUM   | 1 = DEV_NUMBER contains data and the DEV_NUMBER is defined in the configuration                                                                                                              |  |  |
|                                                                                                                                                                                                                                                                                                                                                                               |      | ...1 .... |     | PATHTEST       | 1 = PATHTEST data is available                                                                                                                                                               |  |  |
|                                                                                                                                                                                                                                                                                                                                                                               |      | .... 1... |     | VALID_CUNUM    | 1 = CU_NUMBER contains data and the CU_NUMBER is defined in the configuration                                                                                                                |  |  |
|                                                                                                                                                                                                                                                                                                                                                                               |      | .... .1.. |     | CU_ISA_CF      | 1 = CU in this row is a coupling facility so PTOK mapping is for ND (when 0, PTOK mapping is for NED)                                                                                        |  |  |
|                                                                                                                                                                                                                                                                                                                                                                               |      | .... ..1. |     | VALID_CHP      | 1 = CHPID contains a value that is defined in the configuration                                                                                                                              |  |  |
|                                                                                                                                                                                                                                                                                                                                                                               |      | .... ...1 |     | CASCADED_SW    | 1 = If PORTOUT is valid it represents the output port on the destination switch                                                                                                              |  |  |
| The following indicates whether the current row is to be processed when this table is used as the input (to a Query Entity command)                                                                                                                                                                                                                                           |      |           |     |                |                                                                                                                                                                                              |  |  |
| 68                                                                                                                                                                                                                                                                                                                                                                            | (44) | UNSIGNED  | 1   | CODE           | For Query Entity command. This space is used to tell IHV whether to operate on the given row.<br>0 = Ignore this row<br>1 = Operate on is row<br>2-255 = Reserved (row ignored if specified) |  |  |
| 69                                                                                                                                                                                                                                                                                                                                                                            | (45) | CHARACTER | 1   | *              | Workarea for internals                                                                                                                                                                       |  |  |
| The following bits describe the data validity of the destination (cascaded) switch                                                                                                                                                                                                                                                                                            |      |           |     |                |                                                                                                                                                                                              |  |  |
| 70                                                                                                                                                                                                                                                                                                                                                                            | (46) | BITSTRING | 1   | STATBITS2      | Indicate data validity                                                                                                                                                                       |  |  |
|                                                                                                                                                                                                                                                                                                                                                                               |      | 1... .... |     | VALID_DEST_LSN | 1 = LSN of destination switch is valid                                                                                                                                                       |  |  |
|                                                                                                                                                                                                                                                                                                                                                                               |      | .1.. .... |     | VALID_DEST_SW  | 1 = Device number of destination switch is valid                                                                                                                                             |  |  |
|                                                                                                                                                                                                                                                                                                                                                                               |      | ..11 1111 |     | *              | Reserved                                                                                                                                                                                     |  |  |
| 71                                                                                                                                                                                                                                                                                                                                                                            | (47) | BITSTRING | 1   | *              |                                                                                                                                                                                              |  |  |
|                                                                                                                                                                                                                                                                                                                                                                               |      | 1... .... |     | TCPNAMER       | 1 = APPLNAMER contains the TCP/IP host name                                                                                                                                                  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                               |      | .111 1111 |     | *              | Reserved                                                                                                                                                                                     |  |  |
| 72                                                                                                                                                                                                                                                                                                                                                                            | (48) | CHARACTER | 8   | SYSPLEX        | Sysplex name (blank if none)                                                                                                                                                                 |  |  |
| 80                                                                                                                                                                                                                                                                                                                                                                            | (50) | CHARACTER | 8   | SYSTEM         | System name                                                                                                                                                                                  |  |  |
| 88                                                                                                                                                                                                                                                                                                                                                                            | (58) | CHARACTER | 24  | RESPONDER      | Responder ID                                                                                                                                                                                 |  |  |
| 88                                                                                                                                                                                                                                                                                                                                                                            | (58) | CHARACTER | 8   | APPLNAMER      | Responder host VTAM application ID or TCP/IP host name                                                                                                                                       |  |  |
| 96                                                                                                                                                                                                                                                                                                                                                                            | (60) | CHARACTER | 8   | SYSPLEXR       | Responder host sysplex name                                                                                                                                                                  |  |  |
| 104                                                                                                                                                                                                                                                                                                                                                                           | (68) | CHARACTER | 8   | SYSTEMR        | Responder host system name                                                                                                                                                                   |  |  |
| 112                                                                                                                                                                                                                                                                                                                                                                           | (70) | UNSIGNED  | 4   | RCODE          | Return or reason code for row                                                                                                                                                                |  |  |
| A PATH_AVAIL is returned ONLY when chp/switdevn, chp/cunum or chp/devnum are in the row. In other words, the following commands will return PATH_AVAIL data (when the row contains valid and complete data): Query Relation Host or Chp to Switch (where SWDEVN is set), Query Relation Host or Chp to CntlUnit or Dev, and Query Relation Dev or CU or Switch to Host or Chp |      |           |     |                |                                                                                                                                                                                              |  |  |
| 116                                                                                                                                                                                                                                                                                                                                                                           | (74) | BITSTRING | 4   | PATH_AVAIL     | Last known state of this path from CHSC "Store Sch Path Info" instruction                                                                                                                    |  |  |

Table 11. Output Format for all Query Relation Commands (continued)

| Offset |      |           |     |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------|------|-----------|-----|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dec    | Hex  | Type      | Len | Name(Dim)  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 116    | (74) | BITSTRING | 1   | CHSC_LEVEL | Level (that is, scope) of information:<br><br>'10'x = Error affects entire chp<br>'20'x = Error affects destination link<br>'30'x = Error affects logical path<br>'40'x = Error affects I/O on logical path                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 117    | (75) | BITSTRING | 2   | CHSC_CODE  | Status code and modifier:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 117    | (75) | BITSTRING | 1   | STATCODE   | Status code                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 118    | (76) | BITSTRING | 1   | MODCODE    | Modifier code<br><br>'0000'x = No data available (ESCM value)<br>'00FF'x = Available, operational last time used<br><br>'1010'x = Chpid type does not match hardware type<br>'1020'x = Serial CTC feature not installed<br>'1030'x = ESCON chp connected to ESCON chp (defn err)<br>'1040'x = SCTC connected to ESCON CU<br>'1050'x = Non-CVC connected to converter<br>'1060'x = CVC channel without converter<br>'1070'x = CNC/multiple CU connection with no ESCD<br>'1080'x = No CU link address defined<br>'1090'x = Duplicate link address with port and CU<br>'10A0'x = Msg facility channel connected to another msg facility channel<br>'10C0'x = Buffer sizes incompatible between msg facility channel and msg-processor intersystem channel<br>'10xx'x = Path in definition error, no further info<br>'2010'x = Chpid not configured online<br>'2020'x = Chpid is in check stop state<br>'2030'x = Chpid is in permanent error<br>'20xx'x = Chpid is unavailable<br>'30FF'x = Wrap block is installed<br>'40FF'x = Chpid is in terminal state<br>'5010'x = Loss of signal or sync<br>'5020'x = Not-op sequence recognized<br>'5030'x = Sequence timeout<br>'5040'x = Illegal sequence received<br>'50xx'x = Link failure detected<br>'60FF'x = In offline reception state<br>'7010'x = Port reject—address invalid<br>'7011'x = Undefined destination error<br>'7012'x = Destination port malfunction<br>'7013'x = Port intervention required<br>'70xx'x = Port reject (when no other applies)<br>'8001'x = Link reject—transmit error<br>'8005'x = Link reject—dest. address invalid or error<br>'8007'x = Reserved field error<br>'8008'x = Unrecognized link control function<br>'8009'x = Protocol error<br>'800A'x = ALA error<br>'800B'x = Unrecognized device level<br>'80xx'x = Link level reject encountered |

## Common Elements

Table 11. Output Format for all Query Relation Commands (continued)

| Offset                                                                                                                                                                                           |      |           |     |                                            |                                                                                                               |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|-----------|-----|--------------------------------------------|---------------------------------------------------------------------------------------------------------------|
| Dec                                                                                                                                                                                              | Hex  | Type      | Len | Name(Dim)                                  | Description                                                                                                   |
|                                                                                                                                                                                                  |      | '9010'x = |     | Connection error                           |                                                                                                               |
|                                                                                                                                                                                                  |      | '9020'x = |     | Channel detected transmission error        |                                                                                                               |
|                                                                                                                                                                                                  |      | '9030'x = |     | Protocol error                             |                                                                                                               |
|                                                                                                                                                                                                  |      | '9040'x = |     | Destination address invalid                |                                                                                                               |
|                                                                                                                                                                                                  |      | '9050'x = |     | Device level error                         |                                                                                                               |
|                                                                                                                                                                                                  |      | '90xx'x = |     | Channel link level error                   |                                                                                                               |
|                                                                                                                                                                                                  |      | 'A001'x = |     | Pacing parameters error                    |                                                                                                               |
|                                                                                                                                                                                                  |      | 'A002'x = |     | Logical path resource unavailable          |                                                                                                               |
|                                                                                                                                                                                                  |      | 'A004'x = |     | CU image does not exist                    |                                                                                                               |
|                                                                                                                                                                                                  |      | 'A005'x = |     | Logical path precluded at CU               |                                                                                                               |
|                                                                                                                                                                                                  |      | 'A0xx'x = |     | Logical path unavailable                   |                                                                                                               |
|                                                                                                                                                                                                  |      | 'B010'x = |     | CU device initialization in progress       |                                                                                                               |
|                                                                                                                                                                                                  |      | 'B020'x = |     | Link busy last encountered                 |                                                                                                               |
|                                                                                                                                                                                                  |      | 'B030'x = |     | Port busy last encountered                 |                                                                                                               |
|                                                                                                                                                                                                  |      | 'B040'x = |     | Chpid busy last encountered                |                                                                                                               |
|                                                                                                                                                                                                  |      | 'B0xx'x = |     | Path initialization in progress            |                                                                                                               |
|                                                                                                                                                                                                  |      | 'C010'x = |     | Select-in or address exception             |                                                                                                               |
|                                                                                                                                                                                                  |      | 'C0xx'x = |     | SCH path OK but device not operational     |                                                                                                               |
|                                                                                                                                                                                                  |      | 'FFFF'x = |     | Unknown state or no further info available |                                                                                                               |
| SCPSTATE is returned ONLY when a complete path from chpid to device (or switch device) is in the row.                                                                                            |      |           |     |                                            |                                                                                                               |
| 120                                                                                                                                                                                              | (78) | BITSTRING | 1   | SCPSTATE                                   | State of path from SCP                                                                                        |
|                                                                                                                                                                                                  |      | 1... ..   |     | ONLINE                                     | 1 = Path is online to SCP                                                                                     |
|                                                                                                                                                                                                  |      | .1.. ..   |     | OFFLINE                                    | 1 = Path is offline to SCP                                                                                    |
|                                                                                                                                                                                                  |      | ..11 1111 |     | *                                          | Reserved                                                                                                      |
| Destination switch information                                                                                                                                                                   |      |           |     |                                            |                                                                                                               |
| 121                                                                                                                                                                                              | (79) | UNSIGNED  | 1   | DEST_SWCH_LSN                              | LSN of destination switch                                                                                     |
| 122                                                                                                                                                                                              | (7A) | UNSIGNED  | 2   | DEST_SWCH_DEVN                             | Device number of destination switch                                                                           |
| A LPE_STATUS.ESCON is returned whenever there is a chpid in the row. LPE_STATUS.LPE is ONLY returned when the row contains a valid chpid along with valid switch devnum, cunum or device number. |      |           |     |                                            |                                                                                                               |
| 124                                                                                                                                                                                              | (7C) | BITSTRING | 1   | LPE_STATUS                                 | Logical path established indicators                                                                           |
|                                                                                                                                                                                                  |      | 1... ..   |     | LPE_VALID                                  | 1 = This path supports LPEs AND LPE info is valid                                                             |
|                                                                                                                                                                                                  |      | .1.. ..   |     | LPE                                        | 0 = No path established.<br>1 = A logical path is established (CHSC info). LPE_VALIDbit validates this field. |
|                                                                                                                                                                                                  |      | ..11 1111 |     | *                                          | Reserved                                                                                                      |
| A PTMSG is returned ONLY when PATHTEST is specified in the command. If PATHTEST is not specified, binary zeros are returned in this field.                                                       |      |           |     |                                            |                                                                                                               |
| 125                                                                                                                                                                                              | (7D) | CHARACTER | 71  | PTMSG                                      | MVS or ESCM message due to issuing I/O down this path. Valid only when PATHTEST specfd.                       |
| TOKS are returned for every command where the associated (RNUM) item is set. Additionally, CU & DEV physical tokens are refreshed for each row when PATHTEST is specified.                       |      |           |     |                                            |                                                                                                               |
| <b>Note:</b> Logical tokens consisting of 32 bytes of binary zeros denotes that the LTOK is not valid/available.                                                                                 |      |           |     |                                            |                                                                                                               |

Table 11. Output Format for all Query Relation Commands (continued)

| Offset                                                                                                    |       |           |     |           |                                                                                                             |
|-----------------------------------------------------------------------------------------------------------|-------|-----------|-----|-----------|-------------------------------------------------------------------------------------------------------------|
| Dec                                                                                                       | Hex   | Type      | Len | Name(Dim) | Description                                                                                                 |
| 196                                                                                                       | (C4)  | CHARACTER | 224 | TOKS      | Logical/Physical tokens                                                                                     |
| Chpid tokens:                                                                                             |       |           |     |           |                                                                                                             |
| 196                                                                                                       | (C4)  | CHARACTER | 32  | CHP_PTOK  | Actually determined ND<br>Chpids do not have logical tokens                                                 |
| Switch tokens:                                                                                            |       |           |     |           |                                                                                                             |
| For Query Relation Switch-Switch commands, the tokens returned here are for the entity2 (chained) switch. |       |           |     |           |                                                                                                             |
| 228                                                                                                       | (E4)  | CHARACTER | 32  | SWIT_PTOK | Switch NED if switch is OPEN, or<br>PID if switch is not open (or<br>defined?).                             |
| 260                                                                                                       | (104) | CHARACTER | 32  | SWIT_LTOK | Only valid if this switch is defined<br>as a device to this host—then<br>inherited from CU for this switch. |
| Control Unit tokens:                                                                                      |       |           |     |           |                                                                                                             |
| 292                                                                                                       | (124) | CHARACTER | 32  | CU_PTOK   | NED (or ND if CU_ISA_CF)                                                                                    |
| 324                                                                                                       | (144) | CHARACTER | 32  | CU_LTOK   | From HCD                                                                                                    |
| Device tokens:                                                                                            |       |           |     |           |                                                                                                             |
| 356                                                                                                       | (164) | CHARACTER | 32  | DEV_PTOK  | NED                                                                                                         |
| 388                                                                                                       | (184) | CHARACTER | 32  | DEV_LTOK  | From HCD                                                                                                    |
| 420                                                                                                       | (1A4) | CHARACTER | 0   | *         | Reserved (to round)                                                                                         |

## DELETE FILE

### Purpose

Use the DELETE FILE command at the I/O-Ops API to delete a saved switch configuration that is stored at the switch specified in the command. The switch must be allocated to the issuing I/O-Ops.

### Syntax

►►—DELeTe File—*filename*—*swchdevn*—◄◄

### Parameters

#### **filename**

Specify the file name in 1 through 8 valid EBCDIC codes. Valid codes are uppercase alphabetical characters (A-Z), digital characters (0-9), and 2 special character: the underscore (\_) and the hyphen (-). However, do not specify the following file names: AUX, COM $n$  (where  $n=1-4$ ), CON, IPL, LPT $n$  (where  $n=1-3$ ), NUL, or PRN.

#### **swchdevn**

Specify the switch device number in up to 4 hexadecimal digits. The switch must be allocated, or attached, to the issuing I/O-Ops. You can issue the DISPLAY SWITCH command to obtain a list of these switches.

### Usage

You cannot delete the switch IPL file, which is supplied with each IBM Director and is activated automatically when the unit is powered on.

## QUERY ENTITY CHP

### Purpose

Use the QUERY ENTITY CHP command at the API to obtain data about the channel path (Chp) that you specify.

### Query Parameters

►►—Query Entity Chp—| Entity\_Object |—| Scope |—————►►

### Output

The format of the output from QUERY ENTITY CHP is as follows:

Table 12. QUERY ENTITY CHP Output

| Offset                                                                                                                                                                           |  | Dec  | Hex       | Type      | Len | Name(Dim)                        | Description                                 |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|------|-----------|-----------|-----|----------------------------------|---------------------------------------------|
| 0                                                                                                                                                                                |  | (0)  |           | STRUCTURE | *   | QEC                              |                                             |
| The header for all Query Entity/Interface output structures is not listed here. Its size is 80 bytes. For details see "Common Query Entity/Interface Output Header" on page 203. |  |      |           |           |     |                                  |                                             |
| 80                                                                                                                                                                               |  | (50) |           | CHARACTER | 184 | CHPS(*)                          | Individual chp data                         |
| 80                                                                                                                                                                               |  | (50) |           | UNSIGNED  | 1   | CHPID                            | Channel path ID                             |
| 81                                                                                                                                                                               |  | (51) |           | BITSTRING | 1   | STATBITS                         |                                             |
|                                                                                                                                                                                  |  |      | 1... ..   |           |     | VALID_DATA                       | 1 = This chpid is defined on host           |
|                                                                                                                                                                                  |  |      | .111 111. |           |     | *                                | Reserved                                    |
|                                                                                                                                                                                  |  |      | .... ...1 |           |     | TCPNAMER                         | 1 = APPLNAMER contains the TCP/IP host name |
| 82                                                                                                                                                                               |  | (52) |           | CHARACTER | 32  | CHP_PTOK                         | Physical Token                              |
| 82                                                                                                                                                                               |  | (52) |           | CHARACTER | 32  | ND_DET                           | Determined ND: "who am I"                   |
| 114                                                                                                                                                                              |  | (72) |           | CHARACTER | 32  | ND_ATT                           | Attached ND: "who are you"                  |
| 146                                                                                                                                                                              |  | (92) |           | UNSIGNED  | 1   | TYPE                             |                                             |
|                                                                                                                                                                                  |  |      | '00'x     | = UNDEF   |     | Unknown                          |                                             |
|                                                                                                                                                                                  |  |      | '01'x     | = BLOCK   |     | Parallel block multiplex         |                                             |
|                                                                                                                                                                                  |  |      | '02'x     | = BYTE    |     | Parallel byte multiplex          |                                             |
|                                                                                                                                                                                  |  |      | '03'x     | = CNC_P   |     | ESCON point to point             |                                             |
|                                                                                                                                                                                  |  |      | '04'x     | = CNC_?   |     | ESCON switched or point to point |                                             |
|                                                                                                                                                                                  |  |      | '05'x     | = CNC_S   |     | ESCON switched point to point    |                                             |
|                                                                                                                                                                                  |  |      | '06'x     | = CVC     |     | ESCON path to a block converter  |                                             |
|                                                                                                                                                                                  |  |      | '07'x     | = NTV     |     | Native interface                 |                                             |
|                                                                                                                                                                                  |  |      | '08'x     | = CTC_P   |     | CTC point to point               |                                             |
|                                                                                                                                                                                  |  |      | '09'x     | = CTC_S   |     | CTC switched point to point      |                                             |
|                                                                                                                                                                                  |  |      | '0A'x     | = CTC_?   |     | CTC switched or point to point   |                                             |
|                                                                                                                                                                                  |  |      | '0B'x     | = CFS     |     | Coupling facility sender         |                                             |
|                                                                                                                                                                                  |  |      | '0C'x     | = CFR     |     | Coupling facility receiver       |                                             |
|                                                                                                                                                                                  |  |      | '0F'x     | = CBY     |     | ESCON path to a byte converter   |                                             |
|                                                                                                                                                                                  |  |      | '10'x     | = OSE     |     | OSA express                      |                                             |
|                                                                                                                                                                                  |  |      | '11'x     | = OSD     |     | OSA direct express               |                                             |
|                                                                                                                                                                                  |  |      | '12'x     | = OSA     |     | Open systems adapter             |                                             |

Table 12. QUERY ENTITY CHP Output (continued)

| Offset                       |      |              |     |                             |                                                                                                                                                                                      |
|------------------------------|------|--------------|-----|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dec                          | Hex  | Type         | Len | Name(Dim)                   | Description                                                                                                                                                                          |
|                              |      | '13'x = ISD  |     | Internal system device      |                                                                                                                                                                                      |
|                              |      | '16'x = CBS  |     | Cluster bus sender          |                                                                                                                                                                                      |
|                              |      | '17'x = CBR  |     | Cluster bus receiver        |                                                                                                                                                                                      |
|                              |      | '18'x = ICS  |     | Internal coupling sender    |                                                                                                                                                                                      |
|                              |      | '19'x = ICR  |     | Internal coupling receiver  |                                                                                                                                                                                      |
|                              |      | '1A'x = FC   |     | FICON point to point        |                                                                                                                                                                                      |
|                              |      | '1B'x = FC_S |     | FICON switched              |                                                                                                                                                                                      |
|                              |      | '1C'x = FCV  |     | FICON to ESCON bridge       |                                                                                                                                                                                      |
|                              |      | '1D'x = FC_? |     | FICON incomplete            |                                                                                                                                                                                      |
|                              |      | '1E'x = DSD  |     | Direct system device        |                                                                                                                                                                                      |
|                              |      | '1F'x = EIO  |     | Emulated I/O                |                                                                                                                                                                                      |
|                              |      | '21'x = CBP  |     | Integrated cluster bus peer |                                                                                                                                                                                      |
|                              |      | '22'x = CFP  |     | Coupling facility peer      |                                                                                                                                                                                      |
|                              |      | '23'x = ICP  |     | Internal coupling peer      |                                                                                                                                                                                      |
|                              |      | '24'x = IQD  |     | Internal queued direct comm |                                                                                                                                                                                      |
|                              |      | '25'x = FCP  |     | FCP channel                 |                                                                                                                                                                                      |
|                              |      |              |     | Other values are reserved   |                                                                                                                                                                                      |
| 147                          | (93) | CHARACTER    | 1   | TRAITS                      | Chp characteristics                                                                                                                                                                  |
|                              |      | 1... ..      |     | ONLINE                      | 1 = Chpid is operational on this host                                                                                                                                                |
|                              |      | .1.. ..      |     | DCM_MANAGED                 | 1 = Chpid is DCM managed on this host                                                                                                                                                |
|                              |      | ..11 ..      |     | *                           | Reserved                                                                                                                                                                             |
|                              |      | .... 1111    |     | PROTOCOL                    | Interface protocol used:<br>0 = Unspecified<br>1 = LED<br>2 = Laser<br>3 = Laser-1 (shortwave)<br>4 = Laser-2 (shortwave)<br>5 = Laser-3 (longwave)<br>Other values are reserved     |
| <b>Entity Attribute Mask</b> |      |              |     |                             |                                                                                                                                                                                      |
| 148                          | (94) | BITSTRING    | 4   | EAM                         |                                                                                                                                                                                      |
|                              |      | 1111 ..      |     | LOG_CLASS                   | Logical entity classification:<br>0 = Unspecified<br>1 = Host<br>2 = Chpid<br>3 = Switch<br>4 = Control unit<br>5 = Device<br>7 = Ambiguous (CHCU, etc)<br>other values are reserved |

## QUERY ENTITY CHP

Table 12. QUERY ENTITY CHP Output (continued)

| Offset                         |      |           |      |     |              |                                                                                                                                                                                                                                                                                                      |  |
|--------------------------------|------|-----------|------|-----|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| Dec                            | Hex  | Type      |      | Len | Name(Dim)    | Description                                                                                                                                                                                                                                                                                          |  |
|                                |      | ....      | 1111 |     | PHYS_CLASS   | Physical entity classification:<br>0 = Unspecified<br>1 = Host<br>2 = Chpid<br>3 = Switch<br>4 = Control unit<br>5 = Device<br>6 = ESCON mod2 converter<br>7 = Ambiguous (link is static (no ND_ATT) and IODEF is ambiguous)<br>8 = CF internal path (only set for CUs)<br>other values are reserved |  |
| 149                            | (95) | BITSTRING |      | 2   | STATE        | State of the entity                                                                                                                                                                                                                                                                                  |  |
|                                |      | 1...      | .... |     | LOGICAL      | 1 = Entity is logical                                                                                                                                                                                                                                                                                |  |
|                                |      | .1..      | .... |     | P_CURR       | 1 = Entity is physically current                                                                                                                                                                                                                                                                     |  |
|                                |      | ..1.      | .... |     | P_HIST       | 1 = Entity is physical history                                                                                                                                                                                                                                                                       |  |
|                                |      | ...1      | .... |     | LOG_OTHER    | 1 = Logical by another interface                                                                                                                                                                                                                                                                     |  |
|                                |      | ....      | 1... |     | P_OTHER_CURR | 1 = Physical by another interface                                                                                                                                                                                                                                                                    |  |
|                                |      | ....      | .1.. |     | P_OTHER_HIST | 1 = Physical history by other interface                                                                                                                                                                                                                                                              |  |
|                                |      | ....      | ..1. |     | P_INDIRECT   | 1 = The EAM validity was derived from the attached (ND) interfaces from the control unit<br>0 = The EAM validity was obtained from the CU itself (can only be true for opened switches)                                                                                                              |  |
|                                |      | ....      | ...1 |     | P_AMB        | 1 = Physical ambiguous configured on some interface                                                                                                                                                                                                                                                  |  |
| 150                            | (96) | 1...      | .... |     | LOG_AMB      | 1 = Logical ambiguous configured on some interface                                                                                                                                                                                                                                                   |  |
|                                |      | .1..      | .... |     | CLASS_AMB    | 1 = Logical and physical classes are not compatible                                                                                                                                                                                                                                                  |  |
|                                |      | ..11      | 1111 | >>  | *            | Reserved                                                                                                                                                                                                                                                                                             |  |
| End of Entity Attribute Mask   |      |           |      |     |              |                                                                                                                                                                                                                                                                                                      |  |
| Attached Entity Attribute Mask |      |           |      |     |              |                                                                                                                                                                                                                                                                                                      |  |
| 152                            | (98) | BITSTRING |      | 4   | AEAM         |                                                                                                                                                                                                                                                                                                      |  |



Table 12. QUERY ENTITY CHP Output (continued)

| Offset |      |           |      |     |              |                                                                                                                                                                                                                                                                                                      |  |
|--------|------|-----------|------|-----|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| Dec    | Hex  | Type      |      | Len | Name(Dim)    | Description                                                                                                                                                                                                                                                                                          |  |
|        |      | 1111      | .... |     | LOG_CLASS    | Logical entity classification:<br>0 = Unspecified<br>1 = Host<br>2 = Chpid<br>3 = Switch<br>4 = Control unit<br>5 = Device<br>7 = Ambiguous (CHCU, etc)<br>other values are reserved                                                                                                                 |  |
|        |      | ....      | 1111 |     | PHYS_CLASS   | Physical entity classification:<br>0 = Unspecified<br>1 = Host<br>2 = Chpid<br>3 = Switch<br>4 = Control unit<br>5 = Device<br>6 = ESCON mod2 converter<br>7 = Ambiguous (link is static (no ND_ATT) and IODEF is ambiguous)<br>8 = CF internal path (only set for CUs)<br>other values are reserved |  |
| 153    | (99) | BITSTRING |      | 2   | STATE        | State of the entity                                                                                                                                                                                                                                                                                  |  |
|        |      | 1...      | .... |     | LOGICAL      | 1 = Entity is logical                                                                                                                                                                                                                                                                                |  |
|        |      | .1..      | .... |     | P_CURR       | 1 = Entity is physically current                                                                                                                                                                                                                                                                     |  |
|        |      | ..1.      | .... |     | P_HIST       | 1 = Entity is physical history                                                                                                                                                                                                                                                                       |  |
|        |      | ...1      | .... |     | LOG_OTHER    | 1 = Logical by another interface                                                                                                                                                                                                                                                                     |  |
|        |      | ....      | 1... |     | P_OTHER_CURR | 1 = Physical by another interface                                                                                                                                                                                                                                                                    |  |
|        |      | ....      | .1.. |     | P_OTHER_HIST | 1 = Physical history by other interface                                                                                                                                                                                                                                                              |  |
|        |      | ....      | ..1. |     | P_INDIRECT   | 1 = The attached ND for the entity being queried is history but we got the chpid's validity from the chpid's det ND (which is always valid) so the AEAM is marked P_CURR for the (attached) chpid<br>0 = The attached ND and AEAM have the same validity                                             |  |
|        |      | ....      | ...1 |     | P_AMB        | 1 = Physical ambiguous configured on some interface                                                                                                                                                                                                                                                  |  |

## QUERY ENTITY CHP

Table 12. QUERY ENTITY CHP Output (continued)

| Offset                                  |      |              |     |           |                                                                                                                                                                                                                                                                                                                                                                                           |  |  |
|-----------------------------------------|------|--------------|-----|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|
| Dec                                     | Hex  | Type         | Len | Name(Dim) | Description                                                                                                                                                                                                                                                                                                                                                                               |  |  |
| 154                                     | (9A) | 1... ..      |     | LOG_AMB   | 1 = Logical ambiguous configured on some interface                                                                                                                                                                                                                                                                                                                                        |  |  |
|                                         |      | .1.. ..      |     | CLASS_AMB | 1 = Logical and physical classes are not compatible                                                                                                                                                                                                                                                                                                                                       |  |  |
|                                         |      | ..11 1111 >> |     | *         | Reserved                                                                                                                                                                                                                                                                                                                                                                                  |  |  |
| End of extrapolated entity descriptions |      |              |     |           |                                                                                                                                                                                                                                                                                                                                                                                           |  |  |
| 156                                     | (9C) | CHARACTER    | 36  | OTHERS    |                                                                                                                                                                                                                                                                                                                                                                                           |  |  |
| 156                                     | (9C) | CHARACTER    | 32  | ND        | Extrapolation ND. This field is only valid when AEAM.P_OTHER or AEAM.P_OTHER_HIST are set (on). This ND can be expected to contain a value when either:                                                                                                                                                                                                                                   |  |  |
|                                         |      |              |     |           | <ul style="list-style-type: none"> <li>The PID and ND validities differ (and the validity of this thing better be the same as the PID if there is a PID)</li> <li>There is more than 1 physical (only) path to an attached entity and the path that is being queried is not the most valid path. This ND should contain the identity of the more (most) valid (physical) path.</li> </ul> |  |  |
| 156                                     | (9C) | BITSTRING    | 1   | *         |                                                                                                                                                                                                                                                                                                                                                                                           |  |  |
|                                         |      | 111. ....    |     | NDVALID   | Indicates validity of this ND                                                                                                                                                                                                                                                                                                                                                             |  |  |
|                                         |      | ...1 1111    |     | *         | Reserved                                                                                                                                                                                                                                                                                                                                                                                  |  |  |
| 157                                     | (9D) | CHARACTER    | 31  | *         | Rest of ND                                                                                                                                                                                                                                                                                                                                                                                |  |  |
| 188                                     | (BC) | UNSIGNED     | 2   | LOG       | Extrapolated logical ID                                                                                                                                                                                                                                                                                                                                                                   |  |  |
|                                         |      |              |     |           | This is the lowest logically defined config number assigned to the entity. This field is only valid when AEAM.LOG_OTHER is set (on).                                                                                                                                                                                                                                                      |  |  |
| 190                                     | (BE) | CHARACTER    | 2   | *         | Reserved                                                                                                                                                                                                                                                                                                                                                                                  |  |  |
| 192                                     | (C0) | CHARACTER    | 24  | RESPONDER | Responding host ID                                                                                                                                                                                                                                                                                                                                                                        |  |  |
| 192                                     | (C0) | CHARACTER    | 8   | APPLNAMER | VTAM application name or TCP/IP host name                                                                                                                                                                                                                                                                                                                                                 |  |  |
| 200                                     | (C8) | CHARACTER    | 8   | SYSPLXR   | Sysplex name                                                                                                                                                                                                                                                                                                                                                                              |  |  |
| 208                                     | (D0) | CHARACTER    | 8   | SYSTEMR   | System name                                                                                                                                                                                                                                                                                                                                                                               |  |  |
| 216                                     | (D8) | UNSIGNED     | 4   | RCODE     | Row return/reason code                                                                                                                                                                                                                                                                                                                                                                    |  |  |
| 220                                     | (DC) | CHARACTER    | 5   | CHPIDTYP  | Channel type as string                                                                                                                                                                                                                                                                                                                                                                    |  |  |
| 225                                     | (E1) | UNSIGNED     | 1   | CSSID     | Channel subsystem ID                                                                                                                                                                                                                                                                                                                                                                      |  |  |
| 226                                     | (E2) | CHARACTER    | 38  | CHPIDINFO | Resource Information                                                                                                                                                                                                                                                                                                                                                                      |  |  |
| 226                                     | (E2) | CHARACTER    | 32  | IODF_DESC | HCD User description                                                                                                                                                                                                                                                                                                                                                                      |  |  |

Table 12. QUERY ENTITY CHP Output (continued)

| Offset |       |           |     |                                                   |                                                                                                                                     |
|--------|-------|-----------|-----|---------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| Dec    | Hex   | Type      | Len | Name(Dim)                                         | Description                                                                                                                         |
| 258    | (102) | BITSTRING | 1   | *                                                 | Configure state of channel<br>0 = Reserved<br>1 = Online<br>2 = Offline/standby<br>6 = Offline/reserved<br>Reserved (round to byte) |
|        |       | 111. .... |     | CONFIG_STATE                                      |                                                                                                                                     |
|        |       | ...1 1111 |     | *                                                 |                                                                                                                                     |
| 259    | (103) | CHARACTER | 1   | *                                                 | Reserved (round to even)                                                                                                            |
| 260    | (104) | BITSTRING | 4   | ERROR_STATE                                       | Availability information<br>Last known state of this path from CHSC Store SCH Path Information<br>(ERROR_STATE=0 -> no data avail)  |
| 260    | (104) | BITSTRING | 1   | CHSC_LEVEL                                        | Level (that is, scope) of information                                                                                               |
|        |       | '00'x =   |     | No information available (I/O-Ops')               |                                                                                                                                     |
|        |       | '10'x =   |     | Error affects entire chp                          |                                                                                                                                     |
|        |       | '20'x =   |     | Error affects destination link                    |                                                                                                                                     |
|        |       | '30'x =   |     | Error affects logical path                        |                                                                                                                                     |
|        |       | '40'x =   |     | Error affects I/O on the logical path             |                                                                                                                                     |
| 261    | (105) | BITSTRING | 2   | CHSC_CODE                                         | Status code with modifier                                                                                                           |
| 261    | (105) | BITSTRING | 1   | STATCODE                                          | Status code                                                                                                                         |
| 262    | (106) | BITSTRING | 1   | MODCODE                                           | Status modifier value                                                                                                               |
|        |       | '0000'x = |     | No data available (I/O operations value)          |                                                                                                                                     |
|        |       | '00FF'x = |     | Available, operational last time used             |                                                                                                                                     |
|        |       | '1010'x = |     | Chpid type does not match hardware type           |                                                                                                                                     |
|        |       | '1020'x = |     | Serial CTC feature not installed                  |                                                                                                                                     |
|        |       | '1030'x = |     | ESCON chp connected to ESCON chp (definition err) |                                                                                                                                     |
|        |       | '1040'x = |     | SCTC connected to ESCON CU                        |                                                                                                                                     |
|        |       | '1050'x = |     | Non-CVC connected to converter                    |                                                                                                                                     |
|        |       | '1060'x = |     | CVC channel without converter                     |                                                                                                                                     |
|        |       | '1070'x = |     | CNC/multiple CU connection with no ESCD           |                                                                                                                                     |
|        |       | '1080'x = |     | No CU link address defined                        |                                                                                                                                     |
|        |       | '1090'x = |     | Duplicate link address with port and CU           |                                                                                                                                     |
|        |       | '10xx'x = |     | Path in definition error, no further information  |                                                                                                                                     |

## QUERY ENTITY CNTLUNIT

### Purpose

Use the QUERY ENTITY CNTLUNIT command at the API to obtain data about the specified control unit (CU).

### Query Parameters

►►—Query Entity CntlUnit—| Entity\_Object |—| Scope |—————►►

## QUERY ENTITY CNTLUNIT

### Output

The format of the output from QUERY ENTITY CNTLUNIT is as follows:

Table 13. QUERY ENTITY CNTLUNIT Output

| Offset                                                                                                                                                                           |      |           |     |              |                                                                                                                                                                                               |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|-----------|-----|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dec                                                                                                                                                                              | Hex  | Type      | Len | Name(Dim)    | Description                                                                                                                                                                                   |
| 0                                                                                                                                                                                | (0)  | STRUCTURE | *   | QEU          |                                                                                                                                                                                               |
| The header for all Query Entity/Interface output structures is not listed here. Its size is 80 bytes. For details see "Common Query Entity/Interface Output Header" on page 203. |      |           |     |              |                                                                                                                                                                                               |
| 80                                                                                                                                                                               | (50) | CHARACTER | 200 | CUS(*)       | Control unit descriptions                                                                                                                                                                     |
| 80                                                                                                                                                                               | (50) | UNSIGNED  | 2   | CU_NUMBER    | Control unit number                                                                                                                                                                           |
| 82                                                                                                                                                                               | (52) | CHARACTER | 1   | STATBITS     |                                                                                                                                                                                               |
|                                                                                                                                                                                  |      | 1... ..   |     | VALID_DATA   | 1 = This control unit is defined in the IOCDs                                                                                                                                                 |
|                                                                                                                                                                                  |      | .1.. ..   |     | CU_IS_SWITCH | 1 = This control unit is a switch                                                                                                                                                             |
|                                                                                                                                                                                  |      | ..1. .... |     | CU_IS_CF     | 1 = CU is a coupling facility                                                                                                                                                                 |
|                                                                                                                                                                                  |      | ...1 111. |     | *            | Reserved                                                                                                                                                                                      |
|                                                                                                                                                                                  |      | .... ...1 |     | TCPNAMER     | 1 = APPLNAMER contains the TCP/IP host name                                                                                                                                                   |
| 83                                                                                                                                                                               | (53) | UNSIGNED  | 1   | CUADD        | (IOCP) logical address                                                                                                                                                                        |
| Physical (neighbor "who am I") Data                                                                                                                                              |      |           |     |              |                                                                                                                                                                                               |
| 84                                                                                                                                                                               | (54) | CHARACTER | 32  | PID          | CU's DERIVED physical identity (same format as in HDR)                                                                                                                                        |
| 116                                                                                                                                                                              | (74) | CHARACTER | 32  | CU_PTOK      | Physical Token...remaps the NED (when the CU_IS_CF bit is off) or is an ND (when the CU_IS_CF bit is on (=1'b)). See macro IXLMG for definition of the ND when the CU is a coupling facility. |
| 116                                                                                                                                                                              | (74) | CHARACTER | 32  | NED          | Node Element Descriptor—CU's physical ID read from the control unit when this CU is not a coupling facility                                                                                   |
|                                                                                                                                                                                  |      | 11.. ..   |     | NED_VALID    | Validity bits for PTOK=NED<br>0 = Unused (not valid)<br>1 = Reserved<br>2 = Reserved<br>3 = Valid NED                                                                                         |
| 116                                                                                                                                                                              | (74) | CHARACTER | 32  | ND           | Node Descriptor = CF PTOK                                                                                                                                                                     |
|                                                                                                                                                                                  |      | 111. .... |     | ND_VALID     | Validity bits for PTOK=ND<br>0 = Valid, current<br>1 = Valid, not current<br>2 = Not valid                                                                                                    |
| 148                                                                                                                                                                              | (94) | CHARACTER | 4   | *            | Reserved                                                                                                                                                                                      |
| Entity Attribute Mask                                                                                                                                                            |      |           |     |              |                                                                                                                                                                                               |
| 152                                                                                                                                                                              | (98) | BITSTRING | 4   | EAM          |                                                                                                                                                                                               |

Table 13. QUERY ENTITY CNTLUNIT Output (continued)

| Offset |      |           |      |     |              |                                                                                                                                                                                                                                                                                                      |  |
|--------|------|-----------|------|-----|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| Dec    | Hex  | Type      |      | Len | Name(Dim)    | Description                                                                                                                                                                                                                                                                                          |  |
|        |      | 1111      | .... |     | LOG_CLASS    | Logical entity classification:<br>0 = Unspecified<br>1 = Host<br>2 = Chpid<br>3 = Switch<br>4 = Control unit<br>5 = Device<br>7 = Ambiguous (CHCU, etc)<br>other values are reserved                                                                                                                 |  |
|        |      | ....      | 1111 |     | PHYS_CLASS   | Physical entity classification:<br>0 = Unspecified<br>1 = Host<br>2 = Chpid<br>3 = Switch<br>4 = Control unit<br>5 = Device<br>6 = ESCON mod2 converter<br>7 = Ambiguous (link is static (no ND_ATT) and IODEF is ambiguous)<br>8 = CF internal path (only set for CUs)<br>other values are reserved |  |
| 153    | (99) | BITSTRING |      | 2   | STATE        | State of the entity                                                                                                                                                                                                                                                                                  |  |
|        |      | 1...      | .... |     | LOGICAL      | 1 = Entity is logical                                                                                                                                                                                                                                                                                |  |
|        |      | .1..      | .... |     | P_CURR       | 1 = Entity is physically current                                                                                                                                                                                                                                                                     |  |
|        |      | ..1.      | .... |     | P_HIST       | 1 = Entity is physical history                                                                                                                                                                                                                                                                       |  |
|        |      | ...1      | .... |     | LOG_OTHER    | 1 = Logical by another interface                                                                                                                                                                                                                                                                     |  |
|        |      | ....      | 1... |     | P_OTHER_CURR | 1 = Physical by another interface                                                                                                                                                                                                                                                                    |  |
|        |      | ....      | .1.. |     | P_OTHER_HIST | 1 = Physical history by other interface                                                                                                                                                                                                                                                              |  |
|        |      | ....      | ..1. |     | P_INDIRECT   | 1 = The EAM validity was derived from the attached (ND) interfaces from the control unit<br>0 = The EAM validity was obtained from the CU itself (can only be true for opened switches)                                                                                                              |  |
|        |      | ....      | ...1 |     | P_AMB        | 1 = Physical ambiguous configured on some interface                                                                                                                                                                                                                                                  |  |
| 154    | (9A) | 1...      | .... |     | LOG_AMB      | 1 = Logical ambiguous configured on some interface                                                                                                                                                                                                                                                   |  |

## QUERY ENTITY CNTLUNIT

Table 13. QUERY ENTITY CNTLUNIT Output (continued)

| Offset                       |       |           |      |     |             |                                                     |  |
|------------------------------|-------|-----------|------|-----|-------------|-----------------------------------------------------|--|
| Dec                          | Hex   | Type      |      | Len | Name(Dim)   | Description                                         |  |
|                              |       | .1..      | .... |     | CLASS_AMB   | 1 = Logical and physical classes are not compatible |  |
|                              |       | ..11      | 1111 | >>  | *           | Reserved                                            |  |
| End of Entity Attribute Mask |       |           |      |     |             |                                                     |  |
| 156                          | (9C)  | CHARACTER |      | 48  | IODFDATA    | Information from HCD                                |  |
| 156                          | (9C)  | UNSIGNED  |      | 4   | GROUP       | Group class (encoded field)                         |  |
|                              |       |           |      |     |             | 1 = DASD                                            |  |
|                              |       |           |      |     |             | 2 = Tape                                            |  |
|                              |       |           |      |     |             | 3 = Cluster controller                              |  |
|                              |       |           |      |     |             | 4 = Communications controller                       |  |
|                              |       |           |      |     |             | 5 = MICR/OCR                                        |  |
|                              |       |           |      |     |             | 6 = Graphics                                        |  |
|                              |       |           |      |     |             | 7 = Unit record device                              |  |
|                              |       |           |      |     |             | 8 = Card reader/punch                               |  |
|                              |       |           |      |     |             | 9 = Display                                         |  |
|                              |       |           |      |     |             | 10 = Term printer                                   |  |
|                              |       |           |      |     |             | 255 = Other                                         |  |
| 160                          | (A0)  | CHARACTER |      | 8   | UNIT        | Control Unit Type                                   |  |
| 168                          | (A8)  | CHARACTER |      | 4   | MODEL       | Control Unit Model                                  |  |
| 172                          | (AC)  | CHARACTER |      | 32  | DESCRIPTION | HCD user description of this object                 |  |
| 204                          | (CC)  | BITSTRING |      | 1   | SCPSTATE    | For Coupling Facility only                          |  |
|                              |       | 1... ..   |      |     | CONNECTED   | 1 = MVS allows operations                           |  |
|                              |       | .1..      | .... |     | MANAGED     | 1 = MVS policy exists                               |  |
|                              |       | ..1.      | .... |     | AVAILABLE   | 1 = Physical path exists                            |  |
|                              |       | ...1      | .... |     | UNAVAILABLE | 1 = No physical path                                |  |
|                              |       | ....      | 1111 |     | *           | Reserved                                            |  |
| 205                          | (CD)  | CHARACTER |      | 3   | *           | Reserved                                            |  |
| 208                          | (D0)  | CHARACTER |      | 8   | CFNAME      | Coupling Facility name                              |  |
| 216                          | (D8)  | CHARACTER |      | 32  | CU_LTOK     | Logical Token (is binary zeros when not available)  |  |
| 248                          | (F8)  | CHARACTER |      | 24  | RESPONDER   | Responding host ID                                  |  |
| 248                          | (F8)  | CHARACTER |      | 8   | APPLNAMER   | VTAM application name or TCP/IP host name           |  |
| 256                          | (100) | CHARACTER |      | 8   | SYSPLXR     | Sysplex name                                        |  |
| 264                          | (108) | CHARACTER |      | 8   | SYSTEMR     | System name                                         |  |
| 272                          | (110) | UNSIGNED  |      | 4   | RCODE       | Row return/reason code                              |  |
| 276                          | (114) | CHARACTER |      | 8   | *           | Reserved                                            |  |

## QUERY ENTITY DEV

### Purpose

Use the QUERY ENTITY DEV command at the API to obtain data about the specified device.

## Query Parameters



## Output

The format of the output from QUERY ENTITY DEV is as follows:

Table 14. QUERY ENTITY DEV Output

| Offset                                                                                                                                                                           |      |           |     |               |                                                                                                                                                                                                                                                                                                      |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|-----------|-----|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dec                                                                                                                                                                              | Hex  | Type      | Len | Name(Dim)     | Description                                                                                                                                                                                                                                                                                          |
| 0                                                                                                                                                                                | (0)  | STRUCTURE | *   | QED           |                                                                                                                                                                                                                                                                                                      |
| The header for all Query Entity/Interface output structures is not listed here. Its size is 80 bytes. For details see "Common Query Entity/Interface Output Header" on page 203. |      |           |     |               |                                                                                                                                                                                                                                                                                                      |
| 80                                                                                                                                                                               | (50) | CHARACTER | 184 | DEVS(*)       | Individual device data                                                                                                                                                                                                                                                                               |
| 80                                                                                                                                                                               | (50) | UNSIGNED  | 2   | DEV_NUMBER    | Device number                                                                                                                                                                                                                                                                                        |
| 82                                                                                                                                                                               | (52) | CHARACTER | 1   | STATBITS      |                                                                                                                                                                                                                                                                                                      |
|                                                                                                                                                                                  |      | 1... ..   |     | VALID_DATA    | 1 = This device is defined                                                                                                                                                                                                                                                                           |
|                                                                                                                                                                                  |      | .1.. ..   |     | DEV_IS_SWITCH | 1 = This device is a switch                                                                                                                                                                                                                                                                          |
|                                                                                                                                                                                  |      | ..1. .... |     | DEV_IS_CF     | 1 = This device is a coupling facility                                                                                                                                                                                                                                                               |
|                                                                                                                                                                                  |      | ...1 .... |     | SELF_DESCR    | 1 = This device supports self-description                                                                                                                                                                                                                                                            |
|                                                                                                                                                                                  |      | .... 111. |     | *             | Reserved                                                                                                                                                                                                                                                                                             |
|                                                                                                                                                                                  |      | .... ...1 |     | TCPNAMER      | 1 = APPLNAMER contains the TCP/IP host name                                                                                                                                                                                                                                                          |
| Entity Attribute Mask                                                                                                                                                            |      |           |     |               |                                                                                                                                                                                                                                                                                                      |
| 83                                                                                                                                                                               | (53) | BITSTRING | 4   | EAM           |                                                                                                                                                                                                                                                                                                      |
|                                                                                                                                                                                  |      | 1111 .... |     | LOG_CLASS     | Logical entity classification:<br>0 = Unspecified<br>1 = Host<br>2 = Chpid<br>3 = Switch<br>4 = Control unit<br>5 = Device<br>7 = Ambiguous (CHCU, etc)<br>other values are reserved                                                                                                                 |
|                                                                                                                                                                                  |      | .... 1111 |     | PHYS_CLASS    | Physical entity classification:<br>0 = Unspecified<br>1 = Host<br>2 = Chpid<br>3 = Switch<br>4 = Control unit<br>5 = Device<br>6 = ESCON mod2 converter<br>7 = Ambiguous (link is static (no ND_ATT) and IODEF is ambiguous)<br>8 = CF internal path (only set for CUs)<br>other values are reserved |

## QUERY ENTITY DEV

Table 14. QUERY ENTITY DEV Output (continued)

| Offset                       |      |              |     |              |                                                                                             |
|------------------------------|------|--------------|-----|--------------|---------------------------------------------------------------------------------------------|
| Dec                          | Hex  | Type         | Len | Name(Dim)    | Description                                                                                 |
| 84                           | (54) | BITSTRING    | 2   | STATE        | State of the entity                                                                         |
|                              |      | 1... ..      |     | LOGICAL      | 1 = Entity is logical                                                                       |
|                              |      | .1.. ..      |     | P_CURR       | 1 = Entity is physically current                                                            |
|                              |      | ..1. ....    |     | P_HIST       | 1 = Entity is physical history                                                              |
|                              |      | ...1 ....    |     | LOG_OTHER    | 1 = Logical by another interface                                                            |
|                              |      | .... 1...    |     | P_OTHER_CURR | 1 = Physical by another interface                                                           |
|                              |      | .... .1..    |     | P_OTHER_HIST | 1 = Physical history by other interface                                                     |
|                              |      | .... ..1.    |     | P_INDIRECT   | 1 = The EAM validity was derived from the attached (ND) interfaces from the control unit    |
|                              |      |              |     |              | 0 = The EAM validity was obtained from the CU itself (can only be true for opened switches) |
|                              |      |              |     | .... ...1    |                                                                                             |
| 85                           | (55) | 1... ..      |     | LOG_AMB      | 1 = Logical ambiguous configured on some interface                                          |
|                              |      | .1.. ..      |     | CLASS_AMB    | 1 = Logical and physical classes are not compatible                                         |
|                              |      | ..11 1111 >> | *   |              | Reserved                                                                                    |
| End of Entity Attribute Mask |      |              |     |              |                                                                                             |
| 87                           | (57) | CHARACTER    | 20  | *            | Reserved                                                                                    |
| 107                          | (6B) | CHARACTER    | 24  | RESPONDER    | Responding host ID                                                                          |
| 107                          | (6B) | CHARACTER    | 8   | APPLNAMER    | VTAM application name or TCP/IP host name                                                   |
| 115                          | (73) | CHARACTER    | 8   | SYSPLXR      | Sysplex name                                                                                |
| 123                          | (7B) | CHARACTER    | 8   | SYSTEMR      | System name                                                                                 |
| 131                          | (83) | CHARACTER    | 1   | *            | Reserved                                                                                    |
| 132                          | (84) | UNSIGNED     | 4   | RCODE        | Row return/reason code                                                                      |
| 136                          | (88) | CHARACTER    | 32  | DEV_PTOK     | Physical Token                                                                              |
| 136                          | (88) | CHARACTER    | 32  | NED          | Node Element Descriptor                                                                     |
| 168                          | (A8) | CHARACTER    | 6   | VOLSER       | Volume serial ID (DASD only, device NED indicates device type)                              |
| 174                          | (AE) | CHARACTER    | 2   | *            | Reserved                                                                                    |
| 176                          | (B0) | BITSTRING    | 4   | SCPSTATE     | Operating system state                                                                      |
|                              |      | 1... ..      |     | BOXED        | 1 = Boxed                                                                                   |
|                              |      | .1.. ..      |     | NOTREADY     | 1 = Not ready                                                                               |
|                              |      | ..1. ....    |     | BUSY         | 1 = Busy                                                                                    |
|                              |      | ...1 ....    |     | RESERVED     | 1 = Reserved                                                                                |
|                              |      | .... 1...    |     | ALLOCATED    | 1 = Allocated                                                                               |
|                              |      | .... .1..    |     | ONLINE       | 1 = Online                                                                                  |



Table 14. QUERY ENTITY DEV Output (continued)

| Offset |       |              |     |             |                                                     |  |  |
|--------|-------|--------------|-----|-------------|-----------------------------------------------------|--|--|
| Dec    | Hex   | Type         | Len | Name(Dim)   | Description                                         |  |  |
|        |       | .... ..1.    |     | UNLOAD      | 1 = Unload pending                                  |  |  |
|        |       | .... ...1    |     | MOUNT       | 1 = Mount pending                                   |  |  |
| 177    | (B1)  | 1... ....    |     | RESPENDING  | 1 = Reserve pending                                 |  |  |
|        |       | .1.. ....    |     | PENDING     | 1 = Pending offline                                 |  |  |
|        |       | ..1. ....    |     | OFFALLOC    | 1 = Offline—allocated to SCP                        |  |  |
|        |       | ...1 ....    |     | OFFESCM     | 1 = Offline due to I/O operations                   |  |  |
|        |       | .... 1...    |     | OFFCUIR     | 1 = Offline due to CUIR                             |  |  |
|        |       | .... .1..    |     | OFFTAPE     | 1 = Offline due to tape                             |  |  |
|        |       | .... ..1.    |     | OFFHIERCH   | 1 = Offline due to hierarchy reason                 |  |  |
|        |       | .... ...1    |     | OFFOPER     | 1 = Offline due to operator                         |  |  |
| 178    | (B2)  | 1... ....    |     | OFFLINE     | 1 = Offline                                         |  |  |
|        |       | .1.. ....    |     | INUSE       | 1 = Device is in use (message device only)          |  |  |
|        |       | ..1. ....    |     | OPERATIONAL | 1 = Device is operational (message device only)     |  |  |
|        |       | ...1 ....    |     | NOTOP       | 1 = Device is not operational (message device only) |  |  |
|        |       | .... 1...    |     | AUTOSW      | 1 = Device is set autoswitch                        |  |  |
|        |       | .... .111 >> |     | *           | Reserved                                            |  |  |
| 180    | (B4)  | CHARACTER    | 48  | IODFDATA    | HCD information                                     |  |  |
| 180    | (B4)  | UNSIGNED     | 4   | GROUP       | Generic type encode:                                |  |  |
|        |       |              |     |             | 1 = DASD                                            |  |  |
|        |       |              |     |             | 2 = Tape                                            |  |  |
|        |       |              |     |             | 3 = Cluster controller                              |  |  |
|        |       |              |     |             | 4 = Communications controller                       |  |  |
|        |       |              |     |             | 5 = MICR/OCR                                        |  |  |
|        |       |              |     |             | 6 = Graphics                                        |  |  |
|        |       |              |     |             | 7 = Unit record device                              |  |  |
|        |       |              |     |             | 8 = Card reader/punch                               |  |  |
|        |       |              |     |             | 9 = Display                                         |  |  |
|        |       |              |     |             | 10 = Term printer                                   |  |  |
|        |       |              |     |             | 255 = Other                                         |  |  |
| 184    | (B8)  | CHARACTER    | 8   | UNIT        | Unit                                                |  |  |
| 192    | (C0)  | CHARACTER    | 4   | MODEL       | Model                                               |  |  |
| 196    | (C4)  | CHARACTER    | 32  | DESCRIPTION | HCD user description data                           |  |  |
| 228    | (E4)  | CHARACTER    | 32  | DEV_LTOK    | Logical Token (is binary zeros when not available)  |  |  |
| 260    | (104) | CHARACTER    | 4   | *           | Reserved                                            |  |  |

## QUERY ENTITY HOST

### Purpose

Use the QUERY ENTITY HOST command at the API to obtain data about one or more SA z/OS base programs (hosts) that are known to the issuing SA z/OS (primary host).

### Query Parameters

►►—Query Entity Host—| Host\_Entity\_Object | Scope |—————►►

### Output

Output from a QUERY ENTITY command consists of a header, which is identical for each entity with the exception of the "Eye-Catcher" (offset 0), followed by the substructures, which are unique to each type of entity.

The format of the output from QUERY ENTITY HOST is as follows:

Table 15. QUERY ENTITY HOST Output

| Offset                                                                                                                                                                           |  | Dec | Hex  | Type      | Len | Name(Dim)   | Description                                 |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|-----|------|-----------|-----|-------------|---------------------------------------------|
| 0                                                                                                                                                                                |  |     | (0)  | STRUCTURE | *   | QEH         |                                             |
| The header for all Query Entity/Interface output structures is not listed here. Its size is 80 bytes. For details see "Common Query Entity/Interface Output Header" on page 203. |  |     |      |           |     |             |                                             |
| 80                                                                                                                                                                               |  |     | (50) | CHARACTER | 208 | HOSTS(*)    | Individual host data                        |
| 80                                                                                                                                                                               |  |     | (50) | CHARACTER | 8   | APPL_NAME   | VTAM application name                       |
| 82                                                                                                                                                                               |  |     | (52) | CHARACTER | 1   | STATBITS    |                                             |
|                                                                                                                                                                                  |  |     | 1... | ....      |     | VALID_DATA  | 1 = This host is known                      |
|                                                                                                                                                                                  |  |     | .1.. | ....      |     | HOST_OFF    | 1 = This host is reset off                  |
|                                                                                                                                                                                  |  |     | ..1. | ....      |     | IN_SESSION  | For PRIMARY HOST only                       |
|                                                                                                                                                                                  |  |     |      |           |     |             | 1 = I/O operations/VTAM communication ok    |
|                                                                                                                                                                                  |  |     |      |           |     |             | 0 = No I/O operations/VTAM communication    |
|                                                                                                                                                                                  |  |     |      |           |     |             | For SECONDARY HOST only                     |
|                                                                                                                                                                                  |  |     |      |           |     |             | 1 = Appl-to-appl session ok                 |
|                                                                                                                                                                                  |  |     |      |           |     |             | 0 = No session setup                        |
|                                                                                                                                                                                  |  |     | ...1 | ....      |     | BACKING_OUT | 1 = Backout in progress                     |
|                                                                                                                                                                                  |  |     |      |           |     |             | 0 = No backout processing                   |
|                                                                                                                                                                                  |  |     | .... | 1...      |     | IN_SESSION2 | For PRIMARY HOST only:                      |
|                                                                                                                                                                                  |  |     |      |           |     |             | 1 = I/O-Ops/TCP communication ok            |
|                                                                                                                                                                                  |  |     |      |           |     |             | 0 = No I/O-Ops/TCP communication            |
|                                                                                                                                                                                  |  |     | .... | .11.      |     | *           | Reserved                                    |
|                                                                                                                                                                                  |  |     | .... | ...1      |     | TCPNAMER    | 1 = APPLNAMER contains the TCP/IP host name |

Table 15. QUERY ENTITY HOST Output (continued)

| Offset                |      |           |     |              |                                                                                                                                                                                                                                                                                                      |
|-----------------------|------|-----------|-----|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dec                   | Hex  | Type      | Len | Name(Dim)    | Description                                                                                                                                                                                                                                                                                          |
| 89                    | (59) | CHARACTER | 4   | VER_REL      | SA z/OS version and release on this host                                                                                                                                                                                                                                                             |
| 93                    | (5D) | CHARACTER | 32  | PID          | This host PID (same format as in HDR)                                                                                                                                                                                                                                                                |
| Entity Attribute Mask |      |           |     |              |                                                                                                                                                                                                                                                                                                      |
| 125                   | (7D) | BITSTRING | 4   | EAM          |                                                                                                                                                                                                                                                                                                      |
|                       |      | 1111 .... |     | LOG_CLASS    | Logical entity classification:<br>0 = Unspecified<br>1 = Host<br>2 = Chpid<br>3 = Switch<br>4 = Control unit<br>5 = Device<br>7 = Ambiguous (CHCU, etc)<br>other values are reserved                                                                                                                 |
|                       |      | .... 1111 |     | PHYS_CLASS   | Physical entity classification:<br>0 = Unspecified<br>1 = Host<br>2 = Chpid<br>3 = Switch<br>4 = Control unit<br>5 = Device<br>6 = ESCON mod2 converter<br>7 = Ambiguous (link is static (no ND_ATT) and IODEF is ambiguous)<br>8 = CF internal path (only set for CUs)<br>other values are reserved |
| 126                   | (7E) | BITSTRING | 2   | STATE        | State of the entity                                                                                                                                                                                                                                                                                  |
|                       |      | 1... .... |     | LOGICAL      | 1 = Entity is logical                                                                                                                                                                                                                                                                                |
|                       |      | .1.. .... |     | P_CURR       | 1 = Entity is physically current                                                                                                                                                                                                                                                                     |
|                       |      | ..1. .... |     | P_HIST       | 1 = Entity is physical history                                                                                                                                                                                                                                                                       |
|                       |      | ...1 .... |     | LOG_OTHER    | 1 = Logical by another interface                                                                                                                                                                                                                                                                     |
|                       |      | .... 1... |     | P_OTHER_CURR | 1 = Physical by another interface                                                                                                                                                                                                                                                                    |
|                       |      | .... .1.. |     | P_OTHER_HIST | 1 = Physical history by other interface                                                                                                                                                                                                                                                              |
|                       |      | .... ..1. |     | P_INDIRECT   | 1 = The EAM validity was derived from the attached (ND) interfaces from the control unit<br>0 = The EAM validity was obtained from the CU itself (can only be true for opened switches)                                                                                                              |

## QUERY ENTITY HOST

Table 15. QUERY ENTITY HOST Output (continued)

| Offset                       |       |              |     |           |                                                                                                                                                                                     |  |  |
|------------------------------|-------|--------------|-----|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|
| Dec                          | Hex   | Type         | Len | Name(Dim) | Description                                                                                                                                                                         |  |  |
|                              |       | .... 1       |     | P_AMB     | 1 = Physical ambiguous configured on some interface                                                                                                                                 |  |  |
| 127                          | (7F)  | 1... ..      |     | LOG_AMB   | 1 = Logical ambiguous configured on some interface                                                                                                                                  |  |  |
|                              |       | .1.. ..      |     | CLASS_AMB | 1 = Logical and physical classes are not compatible                                                                                                                                 |  |  |
|                              |       | ..11 1111 >> |     | *         | Reserved                                                                                                                                                                            |  |  |
| End of Entity Attribute Mask |       |              |     |           |                                                                                                                                                                                     |  |  |
| 129                          | (81)  | CHARACTER    | 8   | SYSPLEX   | Sysplex name (blank if none)                                                                                                                                                        |  |  |
| 137                          | (89)  | CHARACTER    | 8   | SYSTEM    | System name                                                                                                                                                                         |  |  |
| 145                          | (91)  | CHARACTER    | 24  | RESPONDER | Responding host ID                                                                                                                                                                  |  |  |
| 145                          | (91)  | CHARACTER    | 8   | APPLNAMER | VTAM application name or TCP/IP host name                                                                                                                                           |  |  |
| 153                          | (99)  | CHARACTER    | 8   | SYSPLEXR  | Sysplex name                                                                                                                                                                        |  |  |
| 161                          | (A1)  | CHARACTER    | 8   | SYSTEMR   | System name                                                                                                                                                                         |  |  |
| 169                          | (A9)  | CHARACTER    | 3   | *         | Reserved                                                                                                                                                                            |  |  |
| 172                          | (AC)  | UNSIGNED     | 4   | RCODE     | Row return/reason code                                                                                                                                                              |  |  |
| 176                          | (B0)  | CHARACTER    | 64  | HCD_DATA  | HCD data                                                                                                                                                                            |  |  |
| 176                          | (B0)  | CHARACTER    | 44  | IODF_DSN  | HCD IODF dataset name                                                                                                                                                               |  |  |
| 220                          | (DC)  | UNSIGNED     | 4   | IODFACT   | Hardware and software (CSS/IODF) synch status.<br>Possible values:<br>1 = HW and SW of the active IODF are in sync<br>2 = HW and SW are out of sync<br>3 = No valid HW token exists |  |  |
| 224                          | (E0)  | CHARACTER    | 16  | IODFNAME  | World-wide unique name of the active configuration                                                                                                                                  |  |  |
| 240                          | (F0)  | CHARACTER    | 16  | LOCKOWNER | Process lock owner<br>This field is only valid when this host is the same as the responding host. For other hosts, this field will be blank.                                        |  |  |
| 240                          | (F0)  | CHARACTER    | 8   | SYSTEML   | Application name of user holding process lock                                                                                                                                       |  |  |
| 248                          | (F8)  | CHARACTER    | 8   | USER      | Userid of lock owner                                                                                                                                                                |  |  |
| 256                          | (100) | BITSTRING    | 1   | R3_FNS    | Additional functions installed beyond Release 3                                                                                                                                     |  |  |
|                              |       | 1... ..      |     | SPE1      | 1 = Byte Pacer. OSA, Downlevel MVS, no switch dependency are supported.                                                                                                             |  |  |
|                              |       | .111 1111    |     | *         | Reserved                                                                                                                                                                            |  |  |
| 257                          | (101) | CHARACTER    | 3   | OSLEVEL   | Operating system                                                                                                                                                                    |  |  |
| 257                          | (101) | CHARACTER    | 1   | NAME      | M = MVS<br>V = VM                                                                                                                                                                   |  |  |

Table 15. QUERY ENTITY HOST Output (continued)

| Offset |       |           |     |               |                                                                  |
|--------|-------|-----------|-----|---------------|------------------------------------------------------------------|
| Dec    | Hex   | Type      | Len | Name(Dim)     | Description                                                      |
| 258    | (102) | CHARACTER | 1   | VERSION       | Decimal 0-9                                                      |
| 259    | (103) | CHARACTER | 1   | RELEASE       | Decimal 0-9                                                      |
| 260    | (104) | CHARACTER | 8   | CPUID         | Processor ID (results of STIDP, will be blank when not set)      |
| 268    | (10C) | UNSIGNED  | 2   | CPUADD        | Processor address (results of STADP, will be blank when not set) |
| 270    | (10E) |           | 1   | *             | Reserved                                                         |
| 271    | (10F) | BITSTRING | 1   | COMMFLAGS     | Communication flags                                              |
|        |       | 1... ..   |     | VTAMINSTALLED | 1 = VTAM is installed on this host                               |
|        |       | .1.. ..   |     | TCPINSTALLED  | 1 = TCP/IP is installed on this host                             |
|        |       | ..1. .... |     | IPV6ONLY      | 1 = TCP/IP supports only IPv6 on this host                       |
| 272    | (110) | CHARACTER | 8   | TCPHOSTNAME   | TCP/IP host name                                                 |
| 273    | (118) | CHARACTER | 8   | *             | Reserved                                                         |

## QUERY ENTITY SWITCH

### Purpose

Use the QUERY ENTITY SWITCH command at the API to obtain data about the specified switch.

### Query Parameters

►►—Query Entity Switch—| Entity\_Object | | Scope |—————►►

### Output

The format of the output from QUERY ENTITY SWITCH is as follows:

Table 16. QUERY ENTITY SWITCH Output

| Offset                                                                                                                                                                           |      |           |     |              |                                          |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|-----------|-----|--------------|------------------------------------------|
| Dec                                                                                                                                                                              | Hex  | Type      | Len | Name(Dim)    | Description                              |
| 0                                                                                                                                                                                | (0)  | STRUCTURE | *   | QES          |                                          |
| The header for all Query Entity/Interface output structures is not listed here. Its size is 80 bytes. For details see “Common Query Entity/Interface Output Header” on page 203. |      |           |     |              |                                          |
| 80                                                                                                                                                                               | (50) | CHARACTER | 192 | SWITCHES(*)  | Individual switch data                   |
| 80                                                                                                                                                                               | (50) | UNSIGNED  | 2   | SW_DEVN      | Switch device number                     |
| 82                                                                                                                                                                               | (52) | CHARACTER | 1   | STATBITS     |                                          |
|                                                                                                                                                                                  |      | 1... ..   |     | VALID_DATA   | 1 = This switch is in database           |
|                                                                                                                                                                                  |      | .1.. ..   |     | VALID_SWDEVN | 1 = Switch device number valid           |
|                                                                                                                                                                                  |      | ..1. .... |     | *            | Reserved                                 |
|                                                                                                                                                                                  |      | ...1 .... |     | OPEN         | 1 = Switch is opened (by I/O operations) |
|                                                                                                                                                                                  |      | .... 1... |     | INVALID_LSN  | 1 = LSN is invalid                       |

## QUERY ENTITY SWITCH

Table 16. QUERY ENTITY SWITCH Output (continued)

| Offset                |      |           |     |             |                                                                                                                                                                                                                                                                                                                        |
|-----------------------|------|-----------|-----|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dec                   | Hex  | Type      | Len | Name(Dim)   | Description                                                                                                                                                                                                                                                                                                            |
|                       |      | .... .11. |     | *           | Reserved                                                                                                                                                                                                                                                                                                               |
|                       |      | .... ...1 |     | TCPNAMER    | 1 = APPLNAMER<br>contains the TCP/IP<br>host name                                                                                                                                                                                                                                                                      |
| 83                    | (53) | UNSIGNED  | 1   | LSN         | Logical switch number                                                                                                                                                                                                                                                                                                  |
| 84                    | (54) | CHARACTER | 32  | SW_PTOK     | Physical Token                                                                                                                                                                                                                                                                                                         |
| 84                    | (54) | CHARACTER | 32  | NED         | Node Element Descriptor                                                                                                                                                                                                                                                                                                |
| 116                   | (74) | CHARACTER | 32  | PID         | Unique (physical) ID (same<br>format as in HDR)                                                                                                                                                                                                                                                                        |
| 148                   | (94) | UNSIGNED  | 1   | NPINST      | Number of installed ports                                                                                                                                                                                                                                                                                              |
| 149                   | (95) | UNSIGNED  | 1   | NPIM        | Number of implemented<br>ports (ports ABLE to be<br>installed)                                                                                                                                                                                                                                                         |
| 150                   | (96) | UNSIGNED  | 1   | OP_STATUS   | Operational status<br>0 = Unspecified<br>1 = Not open<br>2 = In contention<br>3 = H/W error<br>4 = System error<br>5 = I/O error<br>6 = Operational<br>7 = Reserved<br>8 = Read only (HCP set)<br>Other values are reserved                                                                                            |
| 151                   | (97) | UNSIGNED  | 1   | A_CUP       | CUP port address                                                                                                                                                                                                                                                                                                       |
| 152                   | (98) | UNSIGNED  | 4   | STATUS_CODE | Error code (if any)                                                                                                                                                                                                                                                                                                    |
| Entity Attribute Mask |      |           |     |             |                                                                                                                                                                                                                                                                                                                        |
| 156                   | (9C) | BITSTRING | 4   | EAM         |                                                                                                                                                                                                                                                                                                                        |
|                       |      | 1111 .... |     | LOG_CLASS   | Logical entity classification:<br>0 = Unspecified<br>1 = Host<br>2 = Chpid<br>3 = Switch<br>4 = Control unit<br>5 = Device<br>7 = Ambiguous (CHCU,<br>etc)<br>other values are reserved                                                                                                                                |
|                       |      | .... 1111 |     | PHYS_CLASS  | Physical entity<br>classification:<br>0 = Unspecified<br>1 = Host<br>2 = Chpid<br>3 = Switch<br>4 = Control unit<br>5 = Device<br>6 = ESCON mod2<br>converter<br>7 = Ambiguous (link is<br>static (no ND_ATT)<br>and IODEF is<br>ambiguous)<br>8 = CF internal path (only<br>set for CUs)<br>other values are reserved |

Table 16. QUERY ENTITY SWITCH Output (continued)

| Offset                       |      |              |      |              |                                                                                             |
|------------------------------|------|--------------|------|--------------|---------------------------------------------------------------------------------------------|
| Dec                          | Hex  | Type         | Len  | Name(Dim)    | Description                                                                                 |
| 157                          | (9D) | BITSTRING    | 2    | STATE        | State of the entity                                                                         |
|                              |      | 1... ..      |      | LOGICAL      | 1 = Entity is logical                                                                       |
|                              |      | .1.. ..      |      | P_CURR       | 1 = Entity is physically current                                                            |
|                              |      | ..1. ....    |      | P_HIST       | 1 = Entity is physical history                                                              |
|                              |      | ...1 ....    |      | LOG_OTHER    | 1 = Logical by another interface                                                            |
|                              |      | .... 1...    |      | P_OTHER_CURR | 1 = Physical by another interface                                                           |
|                              |      | .... .1..    |      | P_OTHER_HIST | 1 = Physical history by other interface                                                     |
|                              |      | .... ..1.    |      | P_INDIRECT   | 1 = The EAM validity was derived from the attached (ND) interfaces from the control unit    |
|                              |      |              |      |              | 0 = The EAM validity was obtained from the CU itself (can only be true for opened switches) |
|                              |      |              |      | .... ..1     | P_AMB                                                                                       |
| 158                          | (9E) | 1... ..      |      | LOG_AMB      | 1 = Logical ambiguous configured on some interface                                          |
|                              |      | .1.. ..      |      | CLASS_AMB    | 1 = Logical and physical classes are not compatible                                         |
|                              |      | ..11 1111 >> | *    |              | Reserved                                                                                    |
| End of Entity Attribute Mask |      |              |      |              |                                                                                             |
| 160                          | (A0) | CHARACTER    | 32   | IODFDESC     | HCD's user description of this object                                                       |
| 192                          | (C0) | CHARACTER    | 7    | ECLEVEL      | Hardware EC level                                                                           |
| 199                          | (C7) | UNSIGNED     | 1    | LOWPORT      | Lowest port address on this switch                                                          |
| 200                          | (C8) | BITSTRING    | 4    | SCPSTATE     | Operating system state                                                                      |
|                              |      | 1... ..      |      | BOXED        | 1 = Boxed                                                                                   |
|                              |      | .1.. ..      |      | NOTREADY     | 1 = Not ready                                                                               |
|                              |      | ..1. ....    |      | BUSY         | 1 = Busy                                                                                    |
|                              |      | ...1 ....    |      | RESERVED     | 1 = Reserved                                                                                |
|                              |      | .... 1...    |      | ALLOCATED    | 1 = Allocated                                                                               |
|                              |      | .... .1..    |      | ONLINE       | 1 = Online                                                                                  |
|                              |      | .... ..1.    |      | UNLOAD       | 1 = Unload pending                                                                          |
|                              |      | .... ...1    |      | MOUNT        | 1 = Mount pending                                                                           |
|                              |      | 201          | (C9) | 1... ..      |                                                                                             |
| .1.. ..                      |      |              |      | PENDING      | 1 = Pending offline                                                                         |
| ..1. ....                    |      |              |      | OFFALLOC     | 1 = Offline—allocated to SCP                                                                |
| ...1 ....                    |      |              |      | OFFFESCM     | 1 = Offline due to I/O operations                                                           |
| .... 1...                    |      |              |      | OFFCUIR      | 1 = Offline due to CUIR                                                                     |

## QUERY ENTITY SWITCH

Table 16. QUERY ENTITY SWITCH Output (continued)

| Offset |       |           |         |     |             |                                                                                      |
|--------|-------|-----------|---------|-----|-------------|--------------------------------------------------------------------------------------|
| Dec    | Hex   | Type      |         | Len | Name(Dim)   | Description                                                                          |
|        |       | ....      | .1..    |     | OFFTAPE     | 1 = Offline due to tape                                                              |
|        |       | ....      | ..1.    |     | OFFHIERCH   | 1 = Offline due to hierarchy reason                                                  |
|        |       | ....      | ...1    |     | OFFOPER     | 1 = Offline due to operator                                                          |
| 202    | (CA)  | 1...      | ....    |     | OFFLINE     | 1 = Offline                                                                          |
|        |       | .1..      | ....    |     | INUSE       | 1 = Device is in use (message device only)                                           |
|        |       | ..1.      | ....    |     | OPERATIONAL | 1 = Device is operational (message device only)                                      |
|        |       | ...1      | ....    |     | NOTOP       | 1 = Device is not operational (message device only)                                  |
|        |       | ....      | 1...    |     | AUTOSW      | 1 = Device is set autoswitch                                                         |
|        |       | ....      | .111 >> |     | *           | Reserved                                                                             |
| 204    | (CC)  | CHARACTER |         | 32  | SW_LTOK     | Logical Token (inherited from CU for this switch—is binary zeros when not available) |
| 236    | (EC)  | CHARACTER |         | 24  | RESPONDER   | Responding host ID                                                                   |
| 236    | (EC)  | CHARACTER |         | 8   | APPLNAMER   | VTAM application name or TCP/IP host name                                            |
| 244    | (F4)  | CHARACTER |         | 8   | SYSPLEXR    | Sysplex name                                                                         |
| 252    | (FC)  | CHARACTER |         | 8   | SYSTEMR     | System name                                                                          |
| 260    | (104) | UNSIGNED  |         | 4   | RCODE       | Row return/reason code                                                               |
| 264    | (108) | CHARACTER |         | 8   | *           | Reserved                                                                             |

## QUERY FILE

### Purpose

Use the QUERY FILE command at the API to retrieve either a single saved switch configuration or a list of all the configurations saved at a switch returned to the caller in the IHVRESP or other user-designated response area. The switch must be allocated, or attached, to the issuing I/O-Ops.

### Syntax

```

▶▶—Query File—*—[filename]—swchdevn—▶▶

```

### Parameters

- \* Specify \* to get a list of the saved switch configurations that are stored at the specified switch.

#### filename

Specify a file name in 1 through 8 valid EBCDIC codes to obtain a single saved configuration. Valid codes are uppercase alphabetical characters (A-Z), digital characters (0-9), and 2 special characters: the underscore (\_) and the hyphen (-).



However, the following file names are not valid: AUX, COM $n$  (where  $n=1-4$ ), CON, LPT $n$  (where  $n=1-3$ ), NUL, or PRN.

**swchdevn**

Specify the switch device number in up to 4 hexadecimal digits. The switch must be allocated, or attached, to the issuing I/O-Ops. You can use the Display Switch command to obtain a list of these switches.

**Usage**

- A maximum number of saved configurations can be stored at a switch. At an IBM Director, you can store 15 saved switch configurations. In addition, the IPL file can be loaded from, and restored at, the IBM Director. The IPL file is supplied with the unit and activated automatically when the Director is powered on.
- You can query the IPL file only if the Active=Save mode is disabled, which means when any changes being made to the active file are not being saved. (For the status of this mode, see the QFILAS field in the format of the output returned from Q F \*)

**Output**

The format of the output from QUERY FILE *filename* is an array of 257 80-byte records. The data is returned in IHVRESP if the caller is a REXX EXEC and in a return area designated by the user if the caller is an assembler program.

- One 80-byte record is returned for each of the 256 ports that can be addressed. The format for these records is the same as given in "Output" on page 248.
- One 80-byte record is returned to identify the file in the following format:

Table 17. QUERY FILE Output of a Particular Configuration

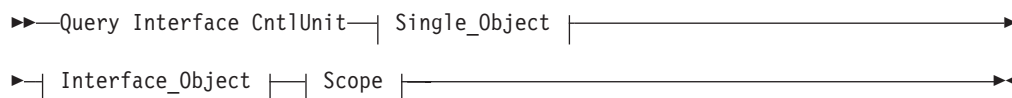
| Offset |      |           |     |           |                                                       |
|--------|------|-----------|-----|-----------|-------------------------------------------------------|
| Dec    | Hex  | Type      | Len | Name(Dim) | Description                                           |
| 0      | (0)  | CHARACTER | 8   | *         | Reserved                                              |
| 8      | (8)  | CHARACTER | 48  | QFILBODY  | Configuration file description (same format as below) |
| 56     | (38) | CHARACTER | 24  | *         | Reserved                                              |

**QUERY INTERFACE CNTLUNIT**

**Purpose**

Use the QUERY INTERFACE CNTLUNIT command at the API to obtain data from the specified control unit regarding its interfaces.

**Query Parameters**



**Parameters**

The QUERY INTERFACE CONTROL UNIT command is designed to work only with ESCON control units because control unit interfaces are *physical* items and

## QUERY INTERFACE CNTLUNIT

only ESCON control units support the architecture to return physical information. No IOCDS pathing information is used to obtain control unit interface responses unless a control specified is a coupling facility control unit. Only IOCDS pathing information is used to obtain the control unit interfaces.

The interface you specify in the command corresponds to the TAG (last 2 bytes, unsigned 2-byte value) field of the node descriptor (ND) associated with the control unit interface.

If the control unit you are querying is a dynamic switch, the interface you specify corresponds to the *port number* of the port that represents the interface.

- For *object\_identifier*, specify the control unit number whose interfaces you want to query.
- For *interface\_identifier*, specify a single physical interface for the specified control unit.
- Specify \* if you want to receive data about all the physical interfaces for the specified control unit. Output array elements are sorted by the DTAG field.  
For *Interface\_identifier* with Range:
- For *lower-upper*, specify an inclusive range of interfaces (or port numbers if the specified control unit is a switch control unit) on the specified control unit. Output array elements are sorted by the DTAG field.
- Specify *lower-\** if you want to receive data about the interfaces from the specified interface to (and including the highest interface. Output array elements are sorted by the DTAG field.

## Output

The format of the output from the QUERY INTERFACE CONTROL UNIT command is as follows:

Table 18. QUERY INTERFACE CNTLUNIT Output

| Offset                                                                                                                                                                           |      |           |     |               |                                             |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|-----------|-----|---------------|---------------------------------------------|
| Dec                                                                                                                                                                              | Hex  | Type      | Len | Name(Dim)     | Description                                 |
| 0                                                                                                                                                                                | (0)  | STRUCTURE | *   | QIU           |                                             |
| The header for all Query Entity/Interface output structures is not listed here. Its size is 80 bytes. For details see "Common Query Entity/Interface Output Header" on page 203. |      |           |     |               |                                             |
| 80                                                                                                                                                                               | (50) | CHARACTER | 312 | INTERFACES(*) | CU interface descriptions                   |
| 80                                                                                                                                                                               | (50) | BITSTRING | 1   |               |                                             |
|                                                                                                                                                                                  |      | 1... ..   |     | VALID_DATA    | 1 = This element contains valid data        |
|                                                                                                                                                                                  |      | .111 111. |     | *             | Reserved                                    |
|                                                                                                                                                                                  |      | .... ...1 |     | TCPNAMER      | 1 = APPLNAMER contains the TCP/IP host name |
| 81                                                                                                                                                                               | (51) | CHARACTER | 7   | *             | Reserved                                    |
| 88                                                                                                                                                                               | (58) | CHARACTER | 32  | ND_DET        | Interface physical identity                 |
| 120                                                                                                                                                                              | (78) | CHARACTER | 32  | ND_ATT        | Interface neighbor physical identity        |
| Attached Entity Attribute Mask                                                                                                                                                   |      |           |     |               |                                             |
| 152                                                                                                                                                                              | (98) | BITSTRING | 4   | AEAM          |                                             |

Table 18. QUERY INTERFACE CNTLUNIT Output (continued)

| Offset |      |           |      |     |              |                                                                                                                                                                                                                                                                                                      |  |
|--------|------|-----------|------|-----|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| Dec    | Hex  | Type      |      | Len | Name(Dim)    | Description                                                                                                                                                                                                                                                                                          |  |
|        |      | 1111      | .... |     | LOG_CLASS    | Logical entity classification:<br>0 = Unspecified<br>1 = Host<br>2 = Chpid<br>3 = Switch<br>4 = Control unit<br>5 = Device<br>7 = Ambiguous (CHCU, etc)<br>other values are reserved                                                                                                                 |  |
|        |      | ....      | 1111 |     | PHYS_CLASS   | Physical entity classification:<br>0 = Unspecified<br>1 = Host<br>2 = Chpid<br>3 = Switch<br>4 = Control unit<br>5 = Device<br>6 = ESCON mod2 converter<br>7 = Ambiguous (link is static (no ND_ATT) and IODEF is ambiguous)<br>8 = CF internal path (only set for CUs)<br>other values are reserved |  |
| 153    | (99) | BITSTRING |      | 2   | STATE        | State of the entity                                                                                                                                                                                                                                                                                  |  |
|        |      | 1...      | .... |     | LOGICAL      | 1 = Entity is logical                                                                                                                                                                                                                                                                                |  |
|        |      | .1..      | .... |     | P_CURR       | 1 = Entity is physically current                                                                                                                                                                                                                                                                     |  |
|        |      | ..1.      | .... |     | P_HIST       | 1 = Entity is physical history                                                                                                                                                                                                                                                                       |  |
|        |      | ...1      | .... |     | LOG_OTHER    | 1 = Logical by another interface                                                                                                                                                                                                                                                                     |  |
|        |      | ....      | 1... |     | P_OTHER_CURR | 1 = Physical by another interface                                                                                                                                                                                                                                                                    |  |
|        |      | ....      | .1.. |     | P_OTHER_HIST | 1 = Physical history by other interface                                                                                                                                                                                                                                                              |  |
|        |      | ....      | ..1. |     | P_INDIRECT   | 1 = The attached ND for the entity being queried is history but we got the chpid's validity from the chpid's det ND (which is always valid) so the AEAM is marked P_CURR for the (attached) chpid<br>0 = The attached ND and AEAM have the same validity                                             |  |
|        |      | ....      | ...1 |     | P_AMB        | 1 = Physical ambiguous configured on some interface                                                                                                                                                                                                                                                  |  |

## QUERY INTERFACE CNTLUNIT

Table 18. QUERY INTERFACE CNTLUNIT Output (continued)

| Offset                                  |      |              |     |           |                                                                                                                                                                                                                                                                                                                                                                                          |  |  |
|-----------------------------------------|------|--------------|-----|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|
| Dec                                     | Hex  | Type         | Len | Name(Dim) | Description                                                                                                                                                                                                                                                                                                                                                                              |  |  |
| 154                                     | (9A) | 1... ..      |     | LOG_AMB   | 1 = Logical ambiguous configured on some interface                                                                                                                                                                                                                                                                                                                                       |  |  |
|                                         |      | .1.. ..      |     | CLASS_AMB | 1 = Logical and physical classes are not compatible                                                                                                                                                                                                                                                                                                                                      |  |  |
|                                         |      | ..11 1111 >> | *   |           | Reserved                                                                                                                                                                                                                                                                                                                                                                                 |  |  |
| End of Attached Entity Attribute Mask   |      |              |     |           |                                                                                                                                                                                                                                                                                                                                                                                          |  |  |
| Extrapolated entity descriptions        |      |              |     |           |                                                                                                                                                                                                                                                                                                                                                                                          |  |  |
| 156                                     | (9C) | CHARACTER    | 36  | OTHERS    |                                                                                                                                                                                                                                                                                                                                                                                          |  |  |
| 156                                     | (9C) | CHARACTER    | 32  | ND        | Extrapolation ND. This field is only valid when AEAM.P_OTHER or AEAM.P_OTHER_HIST are set (on). This ND can be expected to contain a value when either:                                                                                                                                                                                                                                  |  |  |
|                                         |      |              |     |           | <ul style="list-style-type: none"> <li>The PID and ND validities differ (and the validity of this thing better be the same as the PID if there is a PID)</li> <li>There is more than 1 physical (only) path to an attached entity and the path that is being queried is not the most valid path—this ND should contain the identity of the more (most) valid (physical) path.</li> </ul> |  |  |
|                                         |      | 111. ....    |     | NDVALID   | Indicates validity of this ND                                                                                                                                                                                                                                                                                                                                                            |  |  |
|                                         |      | ...1 1111    | *   |           | Not explicitly referenced                                                                                                                                                                                                                                                                                                                                                                |  |  |
| 157                                     | (9D) | CHARACTER    | 31  | *         | Rest of ND                                                                                                                                                                                                                                                                                                                                                                               |  |  |
| 188                                     | (BC) | UNSIGNED     | 2   | LOG       | Extrapolated logical ID This is the lowest logically defined config number assigned to the entity. This field is only valid when AEAM.LOG_OTHER is set (on).                                                                                                                                                                                                                             |  |  |
| 190                                     | (BE) | CHARACTER    | 2   | *         | Reserved                                                                                                                                                                                                                                                                                                                                                                                 |  |  |
| End of extrapolated entity descriptions |      |              |     |           |                                                                                                                                                                                                                                                                                                                                                                                          |  |  |
| 192                                     | (C0) | CHARACTER    | 24  | RESPONDER | Responding host ID                                                                                                                                                                                                                                                                                                                                                                       |  |  |
| 192                                     | (C0) | CHARACTER    | 8   | APPLNAMER | VTAM application name or TCP/IP host name                                                                                                                                                                                                                                                                                                                                                |  |  |
| 200                                     | (C8) | CHARACTER    | 8   | SYSPLXR   | Sysplex name                                                                                                                                                                                                                                                                                                                                                                             |  |  |
| 208                                     | (D0) | CHARACTER    | 8   | SYSTEMR   | System name                                                                                                                                                                                                                                                                                                                                                                              |  |  |
| 216                                     | (D8) | UNSIGNED     | 4   | RCODE     | Row return/reason code                                                                                                                                                                                                                                                                                                                                                                   |  |  |
| Control unit description                |      |              |     |           |                                                                                                                                                                                                                                                                                                                                                                                          |  |  |
| 220                                     | (DC) | CHARACTER    | 168 | CU        |                                                                                                                                                                                                                                                                                                                                                                                          |  |  |
| 220                                     | (DC) | UNSIGNED     | 2   | CU_NUMBER | Control unit number                                                                                                                                                                                                                                                                                                                                                                      |  |  |
| 222                                     | (DE) | CHARACTER    | 1   | STATBITS  |                                                                                                                                                                                                                                                                                                                                                                                          |  |  |

## QUERY INTERFACE CNTLUNIT

Table 18. QUERY INTERFACE CNTLUNIT Output (continued)

| Offset                          |       |           |     |              |             |     |                                                                                                                                                                                           |
|---------------------------------|-------|-----------|-----|--------------|-------------|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dec                             | Hex   | Type      | Len | Name(Dim)    | Description |     |                                                                                                                                                                                           |
|                                 |       | 1... ..   |     | *            |             | 1 = | CU is defined in the IOCDS                                                                                                                                                                |
|                                 |       | .1.. ..   |     | CU_IS_SWITCH |             | 1 = | CU is a switch control unit                                                                                                                                                               |
|                                 |       | ..1. .... |     | CU_IS_CF     |             | 1 = | CU is a coupling facility                                                                                                                                                                 |
|                                 |       | ...1 1111 |     | *            |             |     | Reserved                                                                                                                                                                                  |
| 223                             | (DF)  | UNSIGNED  | 1   | CUADD        |             |     | IOCP logical address                                                                                                                                                                      |
| 224                             | (E0)  | CHARACTER | 32  | PID          |             |     | CU's derived physical identity                                                                                                                                                            |
| 256                             | (100) | CHARACTER | 32  | CU_PTOK      |             |     | Physical Token remaps the NED (when the CU_IS_CF bit is off) or is an ND (when the CU_IS_CF bit is on (=1)). See macro IXLMG for definition of the ND when the CU is a coupling facility. |
| 256                             | (100) | CHARACTER | 32  | NED          |             |     | Node Element Descriptor<br>CU's physical ID read from the control unit when this CU is not a coupling facility                                                                            |
|                                 |       | 11.. .... |     | NED_VALID    |             |     | Validity bits for PTOK=NED<br>0 = Unused (not valid)<br>1 = Reserved<br>2 = Reserved<br>3 = Valid NED                                                                                     |
| 256                             | (100) | CHARACTER | 32  | ND           |             |     | Node Descriptor = CF PTOK                                                                                                                                                                 |
|                                 |       | 111. .... |     | ND_VALID     |             |     | Validity bits for PTOK=ND<br>0 = Valid, current<br>1 = Valid, not current<br>2 = Not valid                                                                                                |
| 288                             | (120) | CHARACTER | 4   | *            |             |     | Reserved                                                                                                                                                                                  |
| End of control unit description |       |           |     |              |             |     |                                                                                                                                                                                           |
| Entity Attribute Mask           |       |           |     |              |             |     |                                                                                                                                                                                           |
| 292                             | (124) | BITSTRING | 4   | EAM          |             |     |                                                                                                                                                                                           |
|                                 |       | 1111 .... |     | LOG_CLASS    |             |     | Logical entity classification:<br>0 = Unspecified<br>1 = Host<br>2 = Chpid<br>3 = Switch<br>4 = Control unit<br>5 = Device<br>7 = Ambiguous (CHCU, etc)<br>other values are reserved      |

## QUERY INTERFACE CNTLUNIT

Table 18. QUERY INTERFACE CNTLUNIT Output (continued)

| Offset                       |       |           |      |     |              |                                                                                                                                                                                                                                                                                                      |  |
|------------------------------|-------|-----------|------|-----|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| Dec                          | Hex   | Type      |      | Len | Name(Dim)    | Description                                                                                                                                                                                                                                                                                          |  |
|                              |       | ....      | 1111 |     | PHYS_CLASS   | Physical entity classification:<br>0 = Unspecified<br>1 = Host<br>2 = Chpid<br>3 = Switch<br>4 = Control unit<br>5 = Device<br>6 = ESCON mod2 converter<br>7 = Ambiguous (link is static (no ND_ATT) and IODEF is ambiguous)<br>8 = CF internal path (only set for CUs)<br>other values are reserved |  |
| 293                          | (125) | BITSTRING |      | 2   | STATE        | State of the entity                                                                                                                                                                                                                                                                                  |  |
|                              |       | 1... ..   |      |     | LOGICAL      | 1 = Entity is logical                                                                                                                                                                                                                                                                                |  |
|                              |       | .1.. ..   |      |     | P_CURR       | 1 = Entity is physically current                                                                                                                                                                                                                                                                     |  |
|                              |       | ..1. .... |      |     | P_HIST       | 1 = Entity is physical history                                                                                                                                                                                                                                                                       |  |
|                              |       | ...1 .... |      |     | LOG_OTHER    | 1 = Logical by another interface                                                                                                                                                                                                                                                                     |  |
|                              |       | .... 1... |      |     | P_OTHER_CURR | 1 = Physical by another interface                                                                                                                                                                                                                                                                    |  |
|                              |       | .... .1.. |      |     | P_OTHER_HIST | 1 = Physical history by other interface                                                                                                                                                                                                                                                              |  |
|                              |       | .... ..1. |      |     | P_INDIRECT   | 1 = The EAM validity was derived from the attached (ND) interfaces from the control unit<br>0 = The EAM validity was obtained from the CU itself (can only be true for opened switches)                                                                                                              |  |
|                              |       | .... ...1 |      |     | P_AMB        | 1 = Physical ambiguous configured on some interface                                                                                                                                                                                                                                                  |  |
| 294                          | (126) | 1... ..   |      |     | LOG_AMB      | 1 = Logical ambiguous configured on some interface                                                                                                                                                                                                                                                   |  |
|                              |       | .1.. ..   |      |     | CLASS_AMB    | 1 = Logical and physical classes are not compatible                                                                                                                                                                                                                                                  |  |
|                              |       | ..11 1111 | >>   | *   |              | Reserved                                                                                                                                                                                                                                                                                             |  |
| End of Entity Attribute Mask |       |           |      |     |              |                                                                                                                                                                                                                                                                                                      |  |
| 296                          | (128) | CHARACTER |      | 48  | IODFDATA     | Information from HCD                                                                                                                                                                                                                                                                                 |  |

Table 18. QUERY INTERFACE CNTLUNIT Output (continued)

| Offset |       |           |     |             |                                                                                                                                                                                                                                                      |
|--------|-------|-----------|-----|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dec    | Hex   | Type      | Len | Name(Dim)   | Description                                                                                                                                                                                                                                          |
| 296    | (128) | UNSIGNED  | 4   | GROUP       | Group class (encoded field)<br>1 = DASD<br>2 = Tape<br>3 = Cluster controller<br>4 = Communications controller<br>5 = MICR/OCR<br>6 = Graphics<br>7 = Unit record device<br>8 = Card reader/punch<br>9 = Display<br>10 = Term printer<br>255 = Other |
| 300    | (12C) | CHARACTER | 8   | UNIT        | Control Unit Type                                                                                                                                                                                                                                    |
| 308    | (134) | CHARACTER | 4   | MODEL       | Control Unit Model                                                                                                                                                                                                                                   |
| 312    | (138) | CHARACTER | 32  | DESCRIPTION | HCD user description of this object                                                                                                                                                                                                                  |
| 344    | (158) | BITSTRING | 1   | SCPSTATE    | For Coupling Facility only                                                                                                                                                                                                                           |
|        |       | 1... ..   |     | CONNECTED   | 1 = MVS allows operations                                                                                                                                                                                                                            |
|        |       | .1.. ..   |     | MANAGED     | 1 = MVS policy exists                                                                                                                                                                                                                                |
|        |       | ..1. ..   |     | AVAILABLE   | 1 = Physical path exists                                                                                                                                                                                                                             |
|        |       | ...1 ..   |     | UNAVAILABLE | 1 = No physical path                                                                                                                                                                                                                                 |
|        |       | .... 1111 |     | *           | Reserved                                                                                                                                                                                                                                             |
| 345    | (159) | CHARACTER | 3   | *           | Reserved                                                                                                                                                                                                                                             |
| 348    | (15C) | CHARACTER | 8   | CFNAME      | Coupling Facility name                                                                                                                                                                                                                               |
| 356    | (164) | CHARACTER | 32  | CU_LTOK     | Logical token (is binary zeros when not available)                                                                                                                                                                                                   |
| 388    | (184) | CHARACTER | 4   | *           | Reserved                                                                                                                                                                                                                                             |

## QUERY INTERFACE SWITCH

### Purpose

Use the QUERY INTERFACE SWITCH command at the API to obtain data about the specified switch regarding its ports.

### Query Parameters

►►—Query Interface Switch—| Single\_Object |—| Interface\_Object |—| Scope |—►►

### Parameters

- For *object\_identifier*, specify the switch device number that you want to receive data about.
- For *interface\_identifier* or \*, a single addressable port on the switch or \* for all the addressable ports on the specified switch. Output array elements are sorted by port address. **Do not enclose the port address in parentheses for this command.**

For *Interface\_identifier* with Range:

## QUERY INTERFACE SWITCH

- For *lower-upper*, specify an inclusive range of port addresses on the specified switch. Output array elements are sorted by port address.
- Specify *lower-\**, if you want to receive data on port addresses, starting with the specified address \* to the highest implemented port address on the specified switch.
- When the CODE value in a row is set to 1, the PORTIN and PORTOUT columns of the table are queried.

## Output

The format of the output from the QUERY INTERFACE SWITCH command is as follows:

Table 19. QUERY INTERFACE SWITCH Output

| Offset                                                                                                                                                                           |      |           |     |                   |                                                              |  |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|-----------|-----|-------------------|--------------------------------------------------------------|--|
| Dec                                                                                                                                                                              | Hex  | Type      | Len | Name(Dim)         | Description                                                  |  |
| 0                                                                                                                                                                                | (0)  | STRUCTURE | *   | QIS               |                                                              |  |
| The header for all Query Entity/Interface output structures is not listed here. Its size is 80 bytes. For details see "Common Query Entity/Interface Output Header" on page 203. |      |           |     |                   |                                                              |  |
| 80                                                                                                                                                                               | (50) | CHARACTER | 360 | PORTS(*)          | Individual port data                                         |  |
| 80                                                                                                                                                                               | (50) | UNSIGNED  | 1   | PORT_NUMBER       | Port Number                                                  |  |
| 81                                                                                                                                                                               | (51) | UNSIGNED  | 1   | PORT_ADDRESS      | Port Address (interface value)                               |  |
| 82                                                                                                                                                                               | (52) | BITSTRING | 1   |                   |                                                              |  |
|                                                                                                                                                                                  |      | 1... ..   |     | VALID_DATA        | 1 = This PORTS element contains valid data                   |  |
|                                                                                                                                                                                  |      | .1.. ..   |     | MID_PORT          | 1 = This port is midport in chain                            |  |
|                                                                                                                                                                                  |      | ..1. .... |     | CHAINED           | 1 = This port is chained                                     |  |
|                                                                                                                                                                                  |      | ...1 1... |     | DCM_STATE         | 0x = This port is not DCM eligible                           |  |
|                                                                                                                                                                                  |      |           |     |                   | 10 = Port is DCM eligible but not allowed for DCM activities |  |
|                                                                                                                                                                                  |      |           |     |                   | 11 = Port is eligible and allowed for DCM activities         |  |
|                                                                                                                                                                                  |      | .... .11. |     | *                 | Reserved                                                     |  |
|                                                                                                                                                                                  |      | .... ...1 |     | TCPNAMER          | 1 = APPLNAMER contains the TCP/IP host name                  |  |
| 83                                                                                                                                                                               | (53) | CHARACTER | 37  | PIB               | Port Information Block                                       |  |
| 83                                                                                                                                                                               | (53) | BITSTRING | 4   | PDB               | Port descriptors                                             |  |
|                                                                                                                                                                                  |      | 1... ..   |     | UNIMPLEMENTED     | 1 = Unimplemented port                                       |  |
|                                                                                                                                                                                  |      | .1.. ..   |     | BLOCKED           | 1 = Port is blocked                                          |  |
|                                                                                                                                                                                  |      | ..1. .... |     | SOME_PDCM_BIT_SET | 1 = At least 1 prohibit                                      |  |
|                                                                                                                                                                                  |      | ...1 .... |     | STATIC            | 1 = This port has static connection                          |  |
|                                                                                                                                                                                  |      | .... 1... |     | *                 | Reserved                                                     |  |



## QUERY INTERFACE SWITCH

Table 19. QUERY INTERFACE SWITCH Output (continued)

| Offset |      |           |     |                   |                                                       |  |  |
|--------|------|-----------|-----|-------------------|-------------------------------------------------------|--|--|
| Dec    | Hex  | Type      | Len | Name(Dim)         | Description                                           |  |  |
|        |      | .... .111 |     | PORT_TECH         | Indicates technology of port H/W                      |  |  |
|        |      |           |     |                   | 0 = Either port not installd or technology is unknown |  |  |
|        |      |           |     |                   | 1 = This is an internal port                          |  |  |
|        |      |           |     |                   | 3 = This port uses LED fiber                          |  |  |
|        |      |           |     |                   | 4 = This port uses LASER fiber technology             |  |  |
|        |      |           |     |                   | Other values are reserved                             |  |  |
| 84     | (54) | 1... .... |     | UNINSTALLED       | 1 = Port is not installed                             |  |  |
|        |      | .1.. .... |     | LINK_FAIL         | 1 = Link failure (hardware fence)                     |  |  |
|        |      | ..1. .... |     | SPARE             | 1 = This is a spare port                              |  |  |
|        |      | ...1 .... |     | OFFLINE           | 1 = Offline (hardware fence)                          |  |  |
|        |      | .... 1... |     | MAINT_MODE        | 1 = In diagnostic (maint) mode                        |  |  |
|        |      | .... .1.. |     | CUP               | 1 = This port is a CUP                                |  |  |
|        |      | .... ..1. |     | SERVICE           | 1 = Service required                                  |  |  |
|        |      | .... ...1 |     | CFG_ERR           | 1 = Invalid (ND) attachment                           |  |  |
| 85     | (55) | 1... .... |     | B_PORT            | 1 = This is a bridge port                             |  |  |
|        |      | .1.. .... |     | PRT_NOTUSABLE     | 1 = Port number not usable                            |  |  |
|        |      | ..1. .... |     | B_PRT_DEGRADED    | 1 = Bridge port degraded                              |  |  |
|        |      | ...1 1... |     | *                 | Reserved                                              |  |  |
|        |      | .... .111 |     | B_PRT_OFFL        | >0 = Bridge port held offline                         |  |  |
| 86     | (56) | 1... .... |     | ERR_THRESHOLD     | 1 = Error threshold exceed                            |  |  |
|        |      | .111 .... |     | PORT_TT           | Transceiver technology valid if PORT_TECH=4:          |  |  |
|        |      |           |     |                   | 0 = Unspecified                                       |  |  |
|        |      |           |     |                   | 1 = GSM                                               |  |  |
|        |      |           |     |                   | 2 = GLS                                               |  |  |
|        |      |           |     |                   | 3 = GLX                                               |  |  |
|        |      | .... 1... |     | *                 | Reserved                                              |  |  |
|        |      | .... .111 |     | PORT_PT           | Protocol type:                                        |  |  |
|        |      |           |     |                   | 0 = ESCON                                             |  |  |
|        |      |           |     |                   | 1 = Reserved                                          |  |  |
|        |      |           |     |                   | 2 = FICON Bridge                                      |  |  |
|        |      |           |     |                   | 3 = FICON Fabric                                      |  |  |
|        |      |           |     |                   | 4 = FICON E-Port                                      |  |  |
|        |      |           |     |                   | 5 = FICON L-Port                                      |  |  |
|        |      |           |     |                   | 6 = FICON G-Port                                      |  |  |
| 87     | (57) | UNSIGNED  | 1   | OTHER_STATIC_PORT | Port that this port is connected to (on same switch)  |  |  |

## QUERY INTERFACE SWITCH

Table 19. QUERY INTERFACE SWITCH Output (continued)

| Offset                         |      |           |     |            |                                                                                                                                                                                                                                                                                                      |
|--------------------------------|------|-----------|-----|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dec                            | Hex  | Type      | Len | Name(Dim)  | Description                                                                                                                                                                                                                                                                                          |
| 88                             | (58) | BITSTRING | 32  | PDCM       | Prohibit Dynamic Connectivity Mask<br>0 = Communication is allowed<br>1 = Communication is not allowed                                                                                                                                                                                               |
| 120                            | (78) | CHARACTER | 24  | LNAME      | Port logical name                                                                                                                                                                                                                                                                                    |
| 144                            | (90) | UNSIGNED  | 1   | IODEF      | Port "type"<br>0 = Unspecified<br>1 = CH (channel)<br>2 = CU<br>3 = CHCU<br>4 = CHCH<br>5 = PC<br>6 = PCCU<br>other values are reserved                                                                                                                                                              |
| 145                            | (91) | CHARACTER | 3   | *          | Reserved                                                                                                                                                                                                                                                                                             |
| 148                            | (94) | CHARACTER | 32  | ND_DET     | Determined ND—"who am I"                                                                                                                                                                                                                                                                             |
| 180                            | (B4) | CHARACTER | 32  | ND_ATT     | Attached ND—"who are you"                                                                                                                                                                                                                                                                            |
| End of switch description      |      |           |     |            |                                                                                                                                                                                                                                                                                                      |
| Attached Entity Attribute Mask |      |           |     |            |                                                                                                                                                                                                                                                                                                      |
| 212                            | (D4) | BITSTRING | 4   | AEAM       | Logical entity classification:<br>0 = Unspecified<br>1 = Host<br>2 = Chpid<br>3 = Switch<br>4 = Control unit<br>5 = Device<br>7 = Ambiguous (CHCU, etc)<br>other values are reserved                                                                                                                 |
|                                |      | 1111 .... |     | LOG_CLASS  |                                                                                                                                                                                                                                                                                                      |
|                                |      | .... 1111 |     | PHYS_CLASS | Physical entity classification:<br>0 = Unspecified<br>1 = Host<br>2 = Chpid<br>3 = Switch<br>4 = Control unit<br>5 = Device<br>6 = ESCON mod2 converter<br>7 = Ambiguous (link is static (no ND_ATT) and IODEF is ambiguous)<br>8 = CF internal path (only set for CUs)<br>other values are reserved |
| 213                            | (D5) | BITSTRING | 2   | STATE      | State of the entity                                                                                                                                                                                                                                                                                  |
|                                |      | 1... .... |     | LOGICAL    | 1 = Entity is logical                                                                                                                                                                                                                                                                                |
|                                |      | .1.. .... |     | P_CURR     | 1 = Entity is physically current                                                                                                                                                                                                                                                                     |

Table 19. QUERY INTERFACE SWITCH Output (continued)

| Offset                                |      |           |         |     |              |                                                                                                                                                                                                                                                          |
|---------------------------------------|------|-----------|---------|-----|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dec                                   | Hex  | Type      |         | Len | Name(Dim)    | Description                                                                                                                                                                                                                                              |
|                                       |      | ..1.      | ....    |     | P_HIST       | 1 = Entity is physical history                                                                                                                                                                                                                           |
|                                       |      | ...1      | ....    |     | LOG_OTHER    | 1 = Logical by another interface                                                                                                                                                                                                                         |
|                                       |      | ....      | 1...    |     | P_OTHER_CURR | 1 = Physical by another interface                                                                                                                                                                                                                        |
|                                       |      | ....      | .1..    |     | P_OTHER_HIST | 1 = Physical history by other interface                                                                                                                                                                                                                  |
|                                       |      | ....      | ..1.    |     | P_INDIRECT   | 1 = The attached ND for the entity being queried is history but we got the chpid's validity from the chpid's det ND (which is always valid) so the AEAM is marked P_CURR for the (attached) chpid<br>0 = The attached ND and AEAM have the same validity |
|                                       |      | ....      | ...1    |     | P_AMB        | 1 = Physical ambiguous configured on some interface                                                                                                                                                                                                      |
| 214                                   | (D6) | 1...      | ....    |     | LOG_AMB      | 1 = Logical ambiguous configured on some interface                                                                                                                                                                                                       |
|                                       |      | .1..      | ....    |     | CLASS_AMB    | 1 = Logical and physical classes are not compatible                                                                                                                                                                                                      |
|                                       |      | ..11      | 1111 >> | *   |              | Reserved                                                                                                                                                                                                                                                 |
| End of Attached Entity Attribute Mask |      |           |         |     |              |                                                                                                                                                                                                                                                          |
| Extrapolated entity descriptions      |      |           |         |     |              |                                                                                                                                                                                                                                                          |
| 216                                   | (D8) | CHARACTER |         | 36  | OTHERS       |                                                                                                                                                                                                                                                          |

## QUERY INTERFACE SWITCH

Table 19. QUERY INTERFACE SWITCH Output (continued)

| Offset                                  |       |           |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----------------------------------------|-------|-----------|-----|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dec                                     | Hex   | Type      | Len | Name(Dim)    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 216                                     | (D8)  | CHARACTER | 32  | ND           | Extrapolation ND. This field is only valid when AEAM.P_OTHER or AEAM.P_OTHER_HIST are set (on). This ND can be expected to contain a value when either: <ul style="list-style-type: none"> <li>• The PID and ND validities differ (and the validity of this thing better be the same as the PID if there is a PID)</li> <li>• There is more than 1 physical (only) path to an attached entity and the path that is being queried is not the most valid path—this ND should contain the identity of the more (most) valid (physical) path.</li> </ul> |
|                                         |       | 111. .... |     | NDVALID      | Indicates validity of this ND                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|                                         |       | ...1 1111 | *   | *            | Not explicitly referenced                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 217                                     | (D9)  | CHARACTER | 31  | *            | Rest of ND                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 248                                     | (F8)  | UNSIGNED  | 2   | LOG          | Extrapolated logical ID<br>This is the lowest logically defined config number assigned to the entity. This field is only valid when AEAM.LOG_OTHER is set (on).                                                                                                                                                                                                                                                                                                                                                                                      |
| 250                                     | (FA)  | CHARACTER | 2   | *            | Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| End of extrapolated entity descriptions |       |           |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 252                                     | (FC)  | CHARACTER | 24  | RESPONDER    | Responding host ID                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 252                                     | (FC)  | CHARACTER | 8   | APPLNAMER    | VTAM application name or TCP/IP host name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 260                                     | (104) | CHARACTER | 8   | SYSPLXR      | Sysplex name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 268                                     | (10C) | CHARACTER | 8   | SYSTEMR      | System name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 276                                     | (114) | UNSIGNED  | 4   | RCODE        | Row return/reason code                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Switch description                      |       |           |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 280                                     | (118) | CHARACTER | 156 | SWITCH       | Switch description (entity1)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 280                                     | (118) | UNSIGNED  | 2   | SW_DEVN      | Switch device number                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 282                                     | (11A) | CHARACTER | 1   | STATBITS     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|                                         |       | 1... .... |     | VALID_DATA   | 1 = This switch is in database                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|                                         |       | .1.. .... |     | VALID_SWDEVN | 1 = Switch device number valid                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|                                         |       | ..1. .... |     | *            | Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|                                         |       | ...1 .... |     | OPEN         | 1 = Switch is opened (by I/O operations)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|                                         |       | .... 1... |     | INVALID_LSN  | 1 = LSN is invalid                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|                                         |       | .... .111 |     | *            | Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

Table 19. QUERY INTERFACE SWITCH Output (continued)

| Offset                    |       |           |     |             |                                                                                                                                                                                                                                                                                                      |
|---------------------------|-------|-----------|-----|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Dec                       | Hex   | Type      | Len | Name(Dim)   | Description                                                                                                                                                                                                                                                                                          |
| 283                       | (11B) | UNSIGNED  | 1   | LSN         | Logical switch number                                                                                                                                                                                                                                                                                |
| 284                       | (11C) | CHARACTER | 32  | SW_PTOK     | Physical Token                                                                                                                                                                                                                                                                                       |
| 284                       | (11C) | CHARACTER | 32  | NED         | Node Element Descriptor                                                                                                                                                                                                                                                                              |
| 316                       | (13C) | CHARACTER | 32  | PID         | Unique (physical) ID (same format as in HDR)                                                                                                                                                                                                                                                         |
| 348                       | (15C) | UNSIGNED  | 1   | NPINST      | Number of installed ports                                                                                                                                                                                                                                                                            |
| 349                       | (15D) | UNSIGNED  | 1   | NPIM        | Number of implemented ports (ports ABLE to be installed)                                                                                                                                                                                                                                             |
| 350                       | (15E) | UNSIGNED  | 1   | OP_STATUS   | Operational status<br>0 = Unspecified<br>1 = Not open<br>2 = In contention<br>3 = H/W error<br>4 = System error<br>5 = I/O error<br>6 = Operational<br>7 = Reserved<br>8 = Read only (HCP set)<br>Other values are reserved                                                                          |
| 351                       | (15F) | UNSIGNED  | 1   | A_CUP       | CUP port address                                                                                                                                                                                                                                                                                     |
| 352                       | (160) | UNSIGNED  | 4   | STATUS_CODE | Error code (if any)                                                                                                                                                                                                                                                                                  |
| End of switch description |       |           |     |             |                                                                                                                                                                                                                                                                                                      |
| Entity Attribute Mask     |       |           |     |             |                                                                                                                                                                                                                                                                                                      |
| 356                       | (164) | BITSTRING | 4   | EAM         |                                                                                                                                                                                                                                                                                                      |
|                           |       | 1111 .... |     | LOG_CLASS   | Logical entity classification:<br>0 = Unspecified<br>1 = Host<br>2 = Chpid<br>3 = Switch<br>4 = Control unit<br>5 = Device<br>7 = Ambiguous (CHCU, etc)<br>other values are reserved                                                                                                                 |
|                           |       | .... 1111 |     | PHYS_CLASS  | Physical entity classification:<br>0 = Unspecified<br>1 = Host<br>2 = Chpid<br>3 = Switch<br>4 = Control unit<br>5 = Device<br>6 = ESCON mod2 converter<br>7 = Ambiguous (link is static (no ND_ATT) and IODEF is ambiguous)<br>8 = CF internal path (only set for CUs)<br>other values are reserved |
| 357                       | (165) | BITSTRING | 2   | STATE       | State of the entity                                                                                                                                                                                                                                                                                  |
|                           |       | 1... .... |     | LOGICAL     | 1 = Entity is logical                                                                                                                                                                                                                                                                                |

## QUERY INTERFACE SWITCH

Table 19. QUERY INTERFACE SWITCH Output (continued)

| Offset                       |       |           |         |     |              |                                                                                             |  |
|------------------------------|-------|-----------|---------|-----|--------------|---------------------------------------------------------------------------------------------|--|
| Dec                          | Hex   | Type      |         | Len | Name(Dim)    | Description                                                                                 |  |
|                              |       | .1..      | ....    |     | P_CURR       | 1 = Entity is physically current                                                            |  |
|                              |       | ..1.      | ....    |     | P_HIST       | 1 = Entity is physical history                                                              |  |
|                              |       | ...1      | ....    |     | LOG_OTHER    | 1 = Logical by another interface                                                            |  |
|                              |       | ....      | 1...    |     | P_OTHER_CURR | 1 = Physical by another interface                                                           |  |
|                              |       | ....      | .1..    |     | P_OTHER_HIST | 1 = Physical history by other interface                                                     |  |
|                              |       | ....      | ..1.    |     | P_INDIRECT   | 1 = The EAM validity was derived from the attached (ND) interfaces from the control unit    |  |
|                              |       |           |         |     |              | 0 = The EAM validity was obtained from the CU itself (can only be true for opened switches) |  |
|                              |       | ....      | ...1    |     | P_AMB        | 1 = Physical ambiguous configured on some interface                                         |  |
| 358                          | (166) | 1...      | ....    |     | LOG_AMB      | 1 = Logical ambiguous configured on some interface                                          |  |
|                              |       | .1..      | ....    |     | CLASS_AMB    | 1 = Logical and physical classes are not compatible                                         |  |
|                              |       | ..11      | 1111 >> | *   |              | Reserved                                                                                    |  |
| End of Entity Attribute Mask |       |           |         |     |              |                                                                                             |  |
| 360                          | (168) | CHARACTER |         | 32  | IODFDESC     | HCD's user description of this object                                                       |  |
| 392                          | (188) | CHARACTER |         | 7   | ECLEVEL      | Hardware EC level                                                                           |  |
| 399                          | (18F) | UNSIGNED  |         | 1   | LOWPORT      | Lowest port address on this switch                                                          |  |
| 400                          | (190) | BITSTRING |         | 4   | SCPSTATE     | Operating system state                                                                      |  |
|                              |       | 1...      | ....    |     | BOXED        | 1 = Boxed                                                                                   |  |
|                              |       | .1..      | ....    |     | NOTREADY     | 1 = Not ready                                                                               |  |
|                              |       | ..1.      | ....    |     | BUSY         | 1 = Busy                                                                                    |  |
|                              |       | ...1      | ....    |     | RESERVED     | 1 = Reserved                                                                                |  |
|                              |       | ....      | 1...    |     | ALLOCATED    | 1 = Allocated                                                                               |  |
|                              |       | ....      | .1..    |     | ONLINE       | 1 = Online                                                                                  |  |
|                              |       | ....      | ..1.    |     | UNLOAD       | 1 = Unload pending                                                                          |  |
|                              |       | ....      | ...1    |     | MOUNT        | 1 = Mount pending                                                                           |  |
| 401                          | (191) | 1...      | ....    |     | RESPENDING   | 1 = Reserve pending                                                                         |  |
|                              |       | .1..      | ....    |     | PENDING      | 1 = Pending offline                                                                         |  |
|                              |       | ..1.      | ....    |     | OFFALLOC     | 1 = Offline—allocated to SCP                                                                |  |
|                              |       | ...1      | ....    |     | OFFFESCM     | 1 = Offline due to I/O operations                                                           |  |
|                              |       | ....      | 1...    |     | OFFCUIR      | 1 = Offline due to CUIR                                                                     |  |
|                              |       | ....      | .1..    |     | OFFTAPE      | 1 = Offline due to tape                                                                     |  |

Table 19. QUERY INTERFACE SWITCH Output (continued)

| Offset |       |              | Len | Name(Dim)   | Description                                                                          |
|--------|-------|--------------|-----|-------------|--------------------------------------------------------------------------------------|
| Dec    | Hex   | Type         |     |             |                                                                                      |
|        |       | .... ..1.    |     | OFFHIERCH   | 1 = Offline due to hierarchy reason                                                  |
|        |       | .... ...1    |     | OFFOPER     | 1 = Offline due to operator                                                          |
| 402    | (192) | 1... ....    |     | OFFLINE     | 1 = Offline                                                                          |
|        |       | .1.. ....    |     | INUSE       | 1 = Device is in use (message device only)                                           |
|        |       | ..1. ....    |     | OPERATIONAL | 1 = Device is operational (message device only)                                      |
|        |       | ...1 ....    |     | NOTOP       | 1 = Device is not operational (message device only)                                  |
|        |       | .... 1... .. |     | AUTOSW      | 1 = Device is set autoswitch                                                         |
|        |       | .... .111 >> |     | *           | Reserved                                                                             |
| 404    | (194) | CHARACTER    | 32  | SW_LTOK     | Logical Token (inherited from CU for this switch—is binary zeros when not available) |
| 436    | (1B4) | CHARACTER    | 4   | *           | Reserved                                                                             |

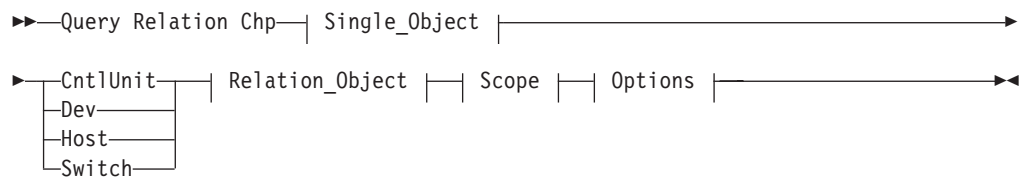
## QUERY RELATION CHP

### Purpose

Use the QUERY RELATION CHP command at the API to obtain data regarding the IOCDS relationship between the two specified entities (objects). Output is based on IOCDS definitions, but it can be influenced by configuration mismatches that have been detected by I/O-Ops.

**Note:** The format of the output from all Query Relation commands is the same. For further information, see “Common Query Relation Output Format” on page 204.

### Query Parameters



### Parameters

- This command returns data about the logical relationships (in IOCDS) between the first entity, which is a single CHPID, and the second entity or entities.
- For a QUERY RELATION command, the first entity (host name) must be known to the issuing I/O-Ops (primary host). The command returns an indication whether the specified CHPID is defined in IOCDS to the issuing I/O-Ops.

## QUERY RELATION CHP

- If you specify switches as the second entity, the command returns an indication whether the CHPID in the first entity has defined paths through the switches.

---

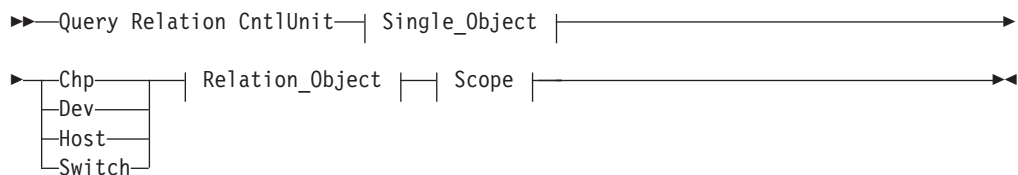
## QUERY RELATION CNTLUNIT

### Purpose

Use the QUERY RELATION CNTLUNIT command at the API to obtain data regarding the IOCDS relationship between the two specified entities (objects). Output is based on IOCDS definitions, but it can be influenced by configuration mismatches that have been detected by I/O-Ops.

**Note:** The format of the output from all Query Relation commands is the same. For further information, see “Common Query Relation Output Format” on page 204.

### Query Parameters



### Parameters

- This command returns data about the relationships between the specified control unit and the second entity in the command.
- If the second entity is Host and the issuing I/O-Ops is included in the parameters, this command returns indications of what CHPIDs have (IOCDS) defined paths to the control unit specified in the first entity.
- If the second entity is Host and a voting I/O-Ops is included in the parameters, this command returns only an indication that the I/O-Ops (secondary host) known to the issuing I/O-Ops. No pathing data can be returned.
- If the second entity is Chp, the command returns indications of whether the specified CHPIDs have (IOCDS) defined paths to the specified control unit for the issuing I/O-Ops (primary host).
- If the second entity is Switch, the command returns indications of whether the control unit specified has (IOCDS) defined paths through the specified switch(es) for the issuing I/O-Ops.
- If the second entity is Dev, the command returns indications of whether the specified control unit has (IOCDS) defined paths through the specified devices for the issuing I/O-Ops.

---

## QUERY RELATION DEV

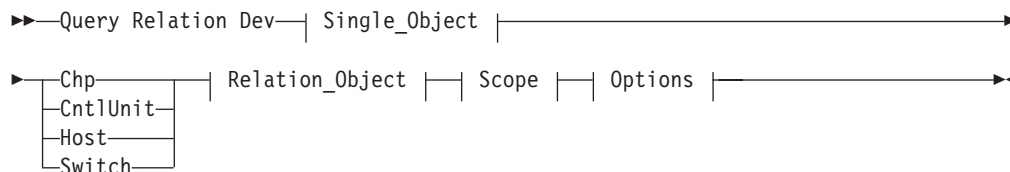
### Purpose

Use the QUERY RELATION DEV command at the API to obtain data regarding the IOCDS relationship between the two specified entities (objects). Output is based on IOCDS definitions, but it can be influenced by configuration mismatches that have been detected by I/O-Ops.



**Note:** The format of the output from all Query Relation commands is the same. For further information, see “Common Query Relation Output Format” on page 204.

## Query Parameters



## Parameters

- This command returns data about the relationships between the specified device and the second entity in the command.
- If the second entity is Host and the issuing I/O-Ops is specified, the command returns indications of whether the device has (IOCDS) defined paths to that host. If a voting I/O-Ops is specified, an indication is returned that the host is known, but pathing information is not available.
- If the second entity is Chp, the command returns indications of whether the specified device has (IOCDS) defined paths to the specified CHPID(s).
- If the second entity is Switch, the command returns indications of whether the control unit specified has (IOCDS) defined paths through the specified switch(es) for the issuing I/O operations.
- If the second entity is CntlUnit, the command returns indications of whether the specified control unit(s) have (IOCDS) defined paths through them to the specified device for the issuing I/O-Ops.

### Notes:

1. When ARRAY is the *Object\_format\_type*, the *Object\_Identifier\_Types* can be mixed and every *Object\_Identifier\_Type* must match the class of the specified *Object\_Type* (all must be I/O\_resources or all must be Hosts). For example, Q E HOST can accept only HOST and XSYS entries in the array.
2. The Array\_header contains the number of elements in the array.
3. PTOK is valid with RANGE but you should be fully aware of PTOK structure. For example, RANGE PTOK could be used to specify all of the serial numbers of a certain type of device. However, certain PTOK values may cause unpredictable results with RANGE.
4. When ARRAY is the *SCOPE\_format\_type*, the *Host\_Object\_Identifier\_Types* can be mixed (HOST and XSYS).
5. If you need to translate a QUERY RELATION command to a new format due to an overflow condition reported by a return code and reason code, you may need to begin the new command with the *last* value that was returned or some pathing information could be lost.

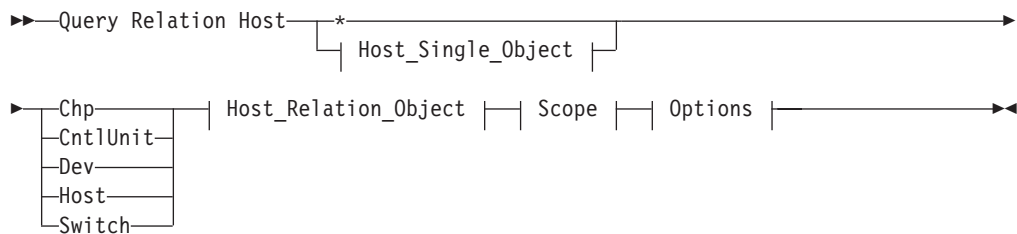
## QUERY RELATION HOST

### Purpose

Use the QUERY RELATION HOST command at the API to obtain data regarding the IOCDS relationship between the two specified entities (objects). Output is based on IOCDS definitions, but it can be influenced by configuration mismatches that have been detected by I/O-Ops.

**Note:** The format of the output from all Query Relation commands is the same. For further information, see “Common Query Relation Output Format” on page 204.

### Query Parameters



### Parameters

- For a QUERY RELATION command, the first entity (host name) must be known to the issuing I/O-Ops (primary host).
- For Q R H S, you can specify any I/O-Ops that participates in vary path consensus processing initiated by the issuing I/O-Ops. However, data indicating CHPID attachments to the switches is returned only for the issuing I/O-Ops

**Notes:**

1. When ARRAY is the *Object\_format\_type*, the *Object\_Identifier\_Types* can be mixed and every *Object\_Identifier\_Type* must match the class of the specified *Object\_Type* (all must be I/O\_resources or all must be Hosts). For example, Q E HOST can accept only HOST and XSYS entries in the array.
2. The Array\_header contains the number of elements in the array.
3. PTOK is valid with RANGE but you should be fully aware of PTOK structure. For example, RANGE PTOK could be used to specify all of the serial numbers of a certain type of device. However, certain PTOK values may cause unpredictable results with RANGE.
4. When ARRAY is the *SCOPE\_format\_type*, the *Host\_Object\_Identifier\_Types* can be mixed (HOST and XSYS).

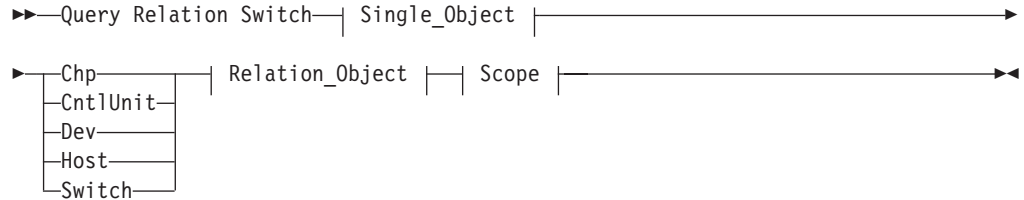
## QUERY RELATION SWITCH

### Purpose

Use the QUERY RELATION SWITCH command at the API to obtain data regarding the IOCDS relationship between the two specified entities (objects). Output is based on IOCDS definitions, but it can be influenced by configuration mismatches that have been detected by I/O-Ops.

**Note:** The format of the output from all Query Relation commands is the same. For further information, see “Common Query Relation Output Format” on page 204.

### Query Parameters



### Parameters

- This command returns data about the relationships between the specified switch and the second entity in the command.
- If you specify the issuing I/O-Ops (host) as the second entity, 1 ROW is returned for each channel that I/O-Ops perceives as being connected to the switch. (If the physical settings at the switch indicate differently from the IOCDS, I/O-Ops "perceives" the physical settings to be accurate.)  
If the switch specifies the destination switch of a cascaded switch pair, the relationship will return one row for each CHP that defines a path to the CUP device of the switch with the following differing information:
  - The output port information shows X'FE' indicating the CUP device of a cascaded switch
- If you specify a voting I/O-Ops (host) as the second entity, only 1 ROW is returned, indicating that the host is able to communicate with, and control, the switch. No CHPIDs are returned, and the incomplete bit is set for that host.
- If you specify CHP as the second entity, the command returns indications of what channel(s) are defined in IOCDS to be attached to the switch. To obtain data on what channels are defined to communicate with a switch, specify Q R CU C, specifying the control unit port, or the Q R D C, specifying the switch device number.
- If you specify Switch as the second entity, the command returns indications:
  - Of what chains have been established with the first entity (ESCON only)
  - Whether both switches build the entry and destination switch of any path defined (FICON only)
- If you specify CntlUnit or Dev as the second entity, the command returns indications whether the specified switch has IOCDS-defined paths through it to the specified control units or devices.

---

## QUERY SWITCH

### Purpose

Use the QUERY SWITCH command at the API to obtain an array of port information blocks (PIBs) and related data from the specified switch.

## Syntax

►►—Query Switch—*swchdevn*—◄◄

## Parameters

### **swchdevn**

Specifies the switch to be queried. The switch must be allocated to, or attached to, the issuing I/O-Ops. Refer to *IBM Tivoli System Automation for z/OS Operator's Commands* for further information about switches.

## Output

The data is presented as an array of 80-byte entries, as shown in Table 20. 256 entries are returned. (The first array is for port address 00.)

Table 20. QUERY SWITCH Output

| Offset |     |           |     |                |                                                           |  |  |
|--------|-----|-----------|-----|----------------|-----------------------------------------------------------|--|--|
| Dec    | Hex | Type      | Len | Name(Dim)      | Description                                               |  |  |
| 0      | (0) | STRUCTURE | 80  | QSWT           |                                                           |  |  |
| 0      | (0) | BITSTRING | 1   | QSWTFLAG1      | Flags byte 1                                              |  |  |
|        |     | 1... ..   |     | QSWTLAST       | End of list indicator                                     |  |  |
|        |     |           |     |                | 0 = More records                                          |  |  |
|        |     |           |     |                | 1 = Last record in array                                  |  |  |
|        |     | .1.. ..   |     | QSWTMDPT       | Midport                                                   |  |  |
|        |     |           |     |                | 1 = This port is the midport of a defined chain           |  |  |
|        |     | ..11 ..   |     | QSWTFORM       | Format ID                                                 |  |  |
|        |     |           |     |                | 0 = Format 0 (original format)                            |  |  |
|        |     | .... 1111 |     | *              | Reserved                                                  |  |  |
| 1      | (1) | BITSTRING | 1   | QSWTFLAG2      | Flags byte 2                                              |  |  |
|        |     | 1111 11.. |     | *              | Reserved                                                  |  |  |
|        |     | .... ..11 |     | QSWT_DCM_STATE | 0x = Port not DCM eligible                                |  |  |
|        |     |           |     |                | 10 = Port DCM eligible but not allowed for DCM activities |  |  |
|        |     |           |     |                | 11 = Port DCM eligible and allowed for DCM activities     |  |  |
| 2      | (2) | UNSIGNED  | 2   | QSWTSWIT       | Switch device number                                      |  |  |
| 4      | (4) | CHARACTER | 48  | QSWTLAIB       | Port information block                                    |  |  |
| 4      | (4) | CHARACTER | 1   | *              | Reserved                                                  |  |  |
| 5      | (5) | UNSIGNED  | 1   | LAIBNUMB       | Port number                                               |  |  |
| 6      | (6) | UNSIGNED  | 1   | LAIBADDR       | Port address                                              |  |  |
| 7      | (7) | CHARACTER | 1   | *              | Reserved                                                  |  |  |
| 8      | (8) | BITSTRING | 4   | LAIBDESC       | Port descriptors                                          |  |  |
|        |     | 1... ..   |     | LAIBUNMP       | Port implementation                                       |  |  |
|        |     |           |     |                | 0 = Port is implemented                                   |  |  |
|        |     |           |     |                | 1 = Port is not implemented                               |  |  |
|        |     | .1.. ..   |     | LAIBFBIT       | Port fence information                                    |  |  |
|        |     |           |     |                | 0 = Port is not blocked                                   |  |  |
|        |     |           |     |                | 1 = Port is blocked                                       |  |  |

Table 20. QUERY SWITCH Output (continued)

| Offset |     | Dec | Hex | Type      | Len | Name(Dim)   | Description                                                                                                                                              |
|--------|-----|-----|-----|-----------|-----|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
|        |     |     |     | ..1. .... |     | LAIBIC      | Prohibit port connection<br>0 = No prohibits for this port<br>1 = Prohibits defined                                                                      |
|        |     |     |     | ..1. .... |     | LAIBLEDEXT  | Port hardware (FICON only)<br>0 = 8 Gbit/sec laser fiber optic<br>1 = 10 Gbit/sec laser fiber optic<br>2 = 16 Gbit/sec laser fiber optic<br>3 = Reserved |
|        |     |     |     | ...1 .... |     | LAIBSBIT    | Port connection (ESCON only)<br>0 = Port is not connected<br>1 = Port is connected                                                                       |
|        |     |     |     | .... 1... |     | *           | Reserved (ESCON only)                                                                                                                                    |
|        |     |     |     | .... .111 |     | LAIBLED     | Port hardware<br>0 = Unspecified<br>1 = Internal<br>2 = Electrical<br>3 = LED fiber optic<br>4 = Laser fiber optic                                       |
| 9      | (9) |     |     | 1... .... |     | LAIBNBIT    | 1 = Not installed                                                                                                                                        |
|        |     |     |     | .1.. .... |     | LAIBLFBIT   | 1 = Link failure                                                                                                                                         |
|        |     |     |     | ..1. .... |     | LAIBSP      | 1 = Swapped port                                                                                                                                         |
|        |     |     |     | ...1 .... |     | LAIBOLBIT   | 1 = Offline                                                                                                                                              |
|        |     |     |     | .... 1... |     | LAIBDMBIT   | 1 = Port in maintenance mode                                                                                                                             |
|        |     |     |     | .... .1.. |     | LAIBCUPBIT  | 1 = This port is a CUP                                                                                                                                   |
|        |     |     |     | .... ..1. |     | LAIBSERVICE | 1 = Service required                                                                                                                                     |
|        |     |     |     | .... ...1 |     | LAIBINVATT  | 1 = Port has an invalid attachment                                                                                                                       |
| 10     | (A) |     |     | 1... .... |     | LAIBBRGPRT  | 1 = This is a bridge port                                                                                                                                |
|        |     |     |     | .1.. .... |     | LAIBPRTNUU  | 1 = Port number not usable                                                                                                                               |
|        |     |     |     | ..1. .... |     | LAIBBPDEG   | 1 = Bridge port degraded                                                                                                                                 |
|        |     |     |     | ...1 1... |     | *           | Reserved                                                                                                                                                 |
|        |     |     |     | .... .111 |     | LAIBBPOFF   | >0 =<br>Bridge port held offline                                                                                                                         |
| 11     | (B) |     |     | 1... .... |     | LAIBETE     | 1 = Error threshold exceeded                                                                                                                             |
|        |     |     |     | .111 .... |     | LAIBTT      | Transeiver technology<br>0 = Unspecified<br>1 = GSM<br>2 = GLS<br>3 = GLX                                                                                |
|        |     |     |     | .... 1... |     | *           | Reserved                                                                                                                                                 |

## QUERY SWITCH

Table 20. QUERY SWITCH Output (continued)

| Offset |      | Type      | Len | Name(Dim) | Description                                                                                                                                     |
|--------|------|-----------|-----|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| Dec    | Hex  |           |     |           |                                                                                                                                                 |
|        |      | .... .111 |     | LAIBPT    | Protocol type:<br>0 = ESCON<br>1 = Reserved<br>2 = FICON Bridge<br>3 = FICON Fabric<br>4 = FICON E-Port<br>5 = FICON L-Port<br>6 = FICON G-Port |
| 12     | (C)  | CHARACTER | 1   | *         | Reserved                                                                                                                                        |
| 13     | (D)  | UNSIGNED  | 1   | LAIBESVR  | Number of ESCON server ports                                                                                                                    |
| 14     | (E)  | UNSIGNED  | 1   | LAIBSADR  | Static connection address                                                                                                                       |
| 15     | (F)  | CHARACTER | 5   | *         | Reserved                                                                                                                                        |
| 20     | (14) | BITSTRING | 32  | LAIBICM   | Port prohibit dynamic connection mask (PDCM)                                                                                                    |
| 52     | (34) | CHARACTER | 24  | QSWTNAME  | Port logical name                                                                                                                               |
| 76     | (4C) | UNSIGNED  | 2   | QSWTCSWIT | Switch device number for chained device                                                                                                         |
| 78     | (4E) | UNSIGNED  | 1   | QSWTCPORT | Chained port address                                                                                                                            |
| 79     | (4F) | CHARACTER | 1   | *         | Reserved                                                                                                                                        |

## Examples

**Note:** If a port is not implemented, only the switch number, port address, and unimplemented bit contain valid data; all other fields are set to binary zeros.

The following sample output shows that port address *F3* has been assigned port name *0500X0600*. It is statically connected to port address *E1* on switch device number *0500*. As one would expect from the port name, port address *F3* is chained to port address *D0* on switch device number *0600*.

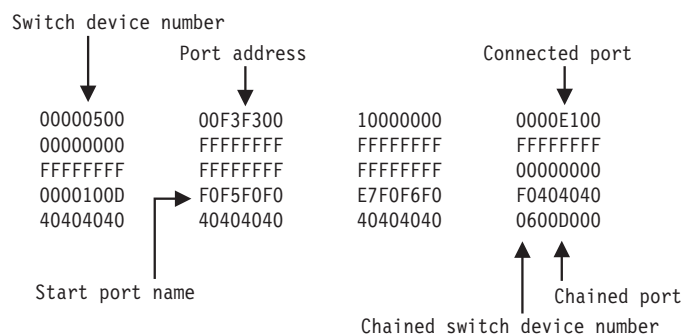


Figure 14. QUERY SWITCH Command - Sample Output.

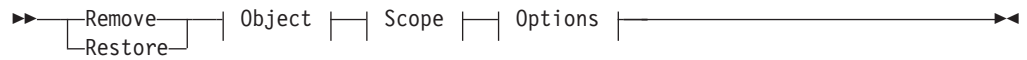
## REMOVE and RESTORE CHP

### Purpose

Use the REMOVE CHP command at the I/O-Ops API to configure a chpid or chpids offline to one or more hosts.

Use the RESTORE CHP command at the I/O-Ops API to configure a chpid or chpids online to one or more hosts.

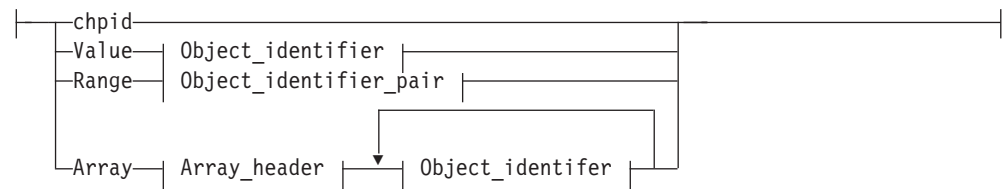
## Syntax



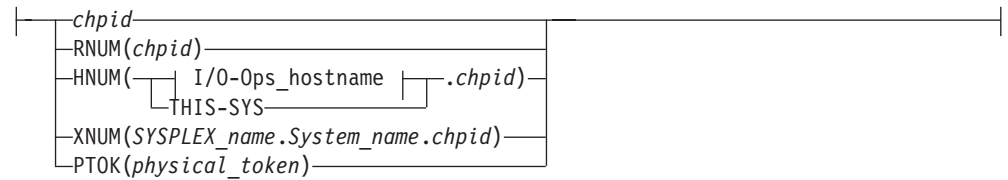
### Object:



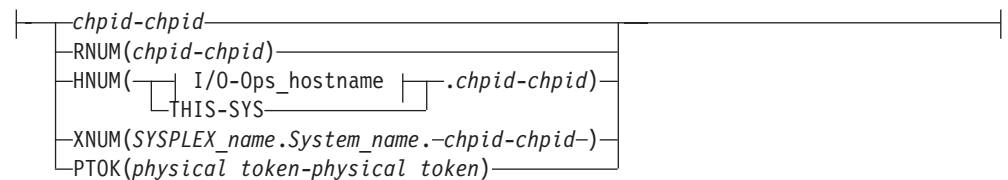
### Object\_format



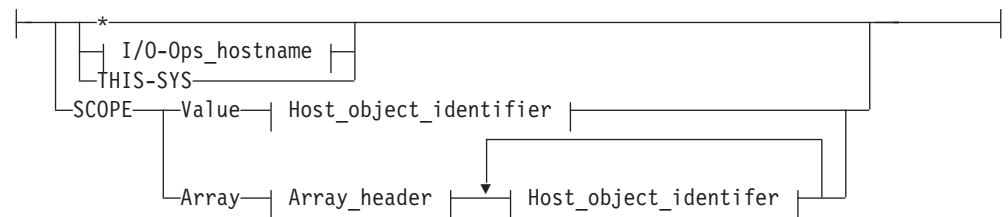
### Object\_identifier:



### Object\_identifier\_pair:

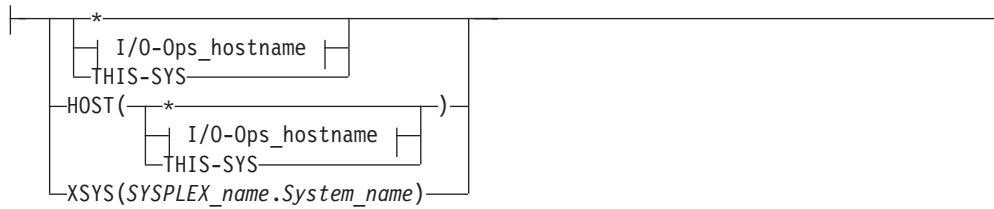


### Scope:

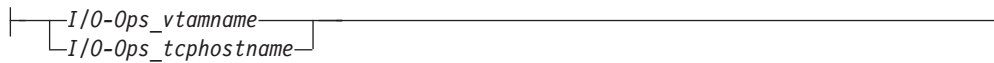


### Host\_object\_identifier:

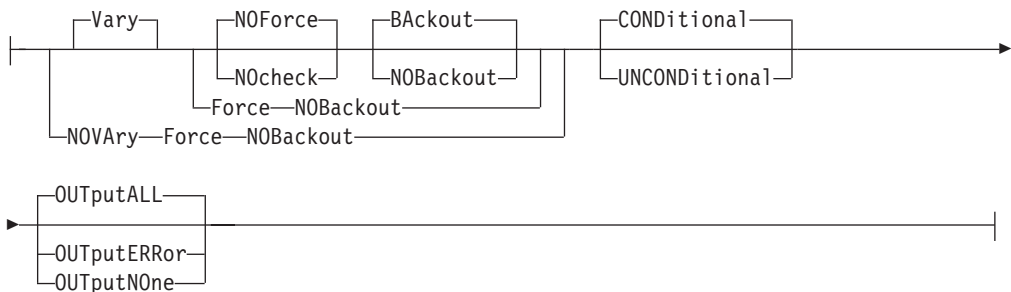
## REMOVE and RESTORE CHP



### I/O-Ops\_hostname:



### Options:



## Parameters

A host identifier type can be one of the following keywords:

### Vary

This is the default option and it indicates that appropriate processing must be done at the host to support the REMOVE and RESTORE CHP commands.

### NOVary

This option is not valid for the REMOVE and RESTORE CHP commands.

### Force

This option says to execute the command in the best manner possible. For example, if one of the specified hosts does not respond, the command is still performed on all other hosts.

### NOForce

This is the default option and indicates that if there is any failure, the command should not continue and a return and reason describing the failure will be returned.

### NOCheck

The NOCheck option overrides the detection of two conditions that would cause the failure of the command under the default NOForce option:

1. Detection of systems in the scope of the command that I/O-Ops is not operating on
2. Detection of downlevel I/O-Ops's operating on systems in the scope of the command

If either of these conditions is detected, a return code of 4 is returned.



**BAckout**

This is the default option and indicates that if any failure is reported by any of the participating systems, any successful REMOVE and RESTORE CHP actions for all the participating host systems will be backed out.

**NOBackout**

This option indicates that if any error condition is detected during the REMOVE and RESTORE CHP processing, I/O-Ops will not attempt to change any REMOVE and RESTORE CHP actions that have been performed.

**CONDitional**

This is the default option for both the REMOVE and RESTORE CHP commands. It indicates that no special configure offline or configure online action should be performed.

**UNCONDitional**

For the REMOVE CHP command, this option puts the specified chpids immediately into pending offline status, even if the chpids are currently active, allocated, or reserved.

For the RESTORE CHP command, this option brings the specified chpids online, even if there are no paths to the chpids, or if the chpids are pending offline and boxed.

**OUTputALL**

This is the default and it allows all results from REMOVE and RESTORE CHPs performed (regardless of return code) to be returned to the API invoker.

**OUTputERRor**

This option allows only error results (REMOVE and RESTORE CHPs that had a return code  $\geq 4$ , plus other errors that occurred during the processing of the command) to be returned to the API invoker.

**OUTputNOne**

This option allows only the return and reason code (no text information) to be returned to the API invoker. If the return code from the command is  $\geq 4$ , a detailed message (IHVC00I, IHVC001I, or IHVC002I) is also returned.

**Notes:**

1. This form, with no keyword, is supported for compatibility with the previously existing syntax of this command.
2. ARRAY does not have a short form for this command (in other multisystem commands A is used as a short form). That is to avoid the need to look ahead in parsing "Remove Chp A..." to distinguish between removing the CHP with ID 'A' and removing an array of CHPs.
3. In this command, THIS-SYS is a means to refer to the primary host (the one that the command is being input to). It is accepted by the primary regardless of whether VTAM is operational or not.
4. When ARRAY is the *Object\_format\_type*, the *Object\_Identifier\_Types* may be mixed (for example, HOST and XSYS), and every *Object\_Identifier\_Type* must be an I/O resource type. For example, an HNUM and an XNUM entry can be in the same array.
5. The Array\_header contains the number of elements in the array.
6. PTOK is valid with RANGE but you should be fully aware of PTOK structure. Certain PTOK values may cause unpredictable results with RANGE.
7. When ARRAY is the *SCOPE\_format\_type*, the *Host\_Object\_Identifier\_Types* may be mixed (for example, HOST and XSYS).

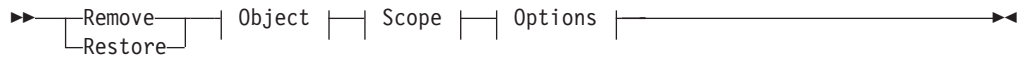
## REMOVE DEV and RESTORE DEV

### Purpose

Use the REMOVE DEV command at the I/O-Ops API to configure a device or devices offline to one or more hosts.

Use the RESTORE DEV command at the I/O-Ops API to configure a device or devices online to one or more hosts.

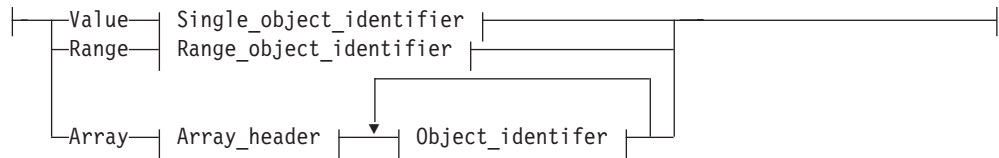
### Syntax



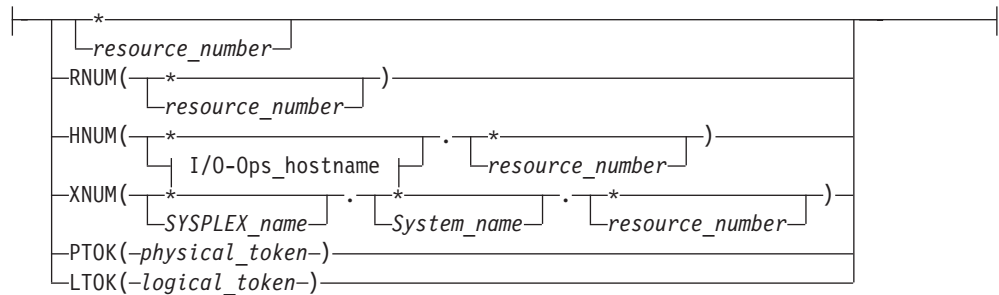
#### Object:



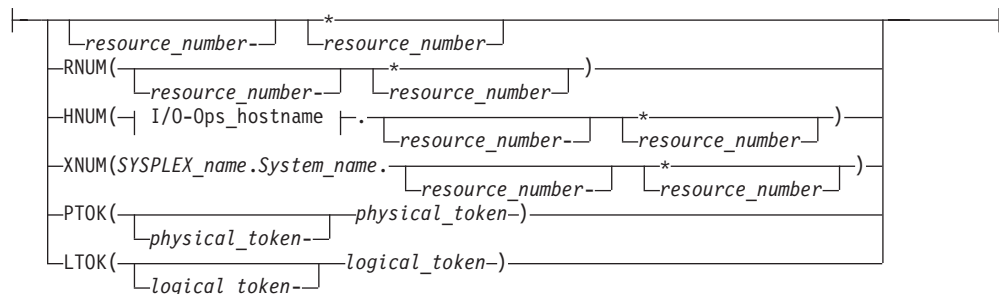
#### Object\_format



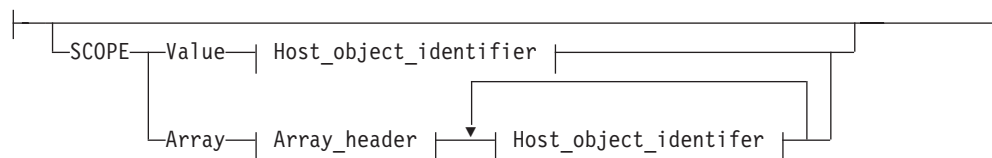
#### Single\_object\_identifier:



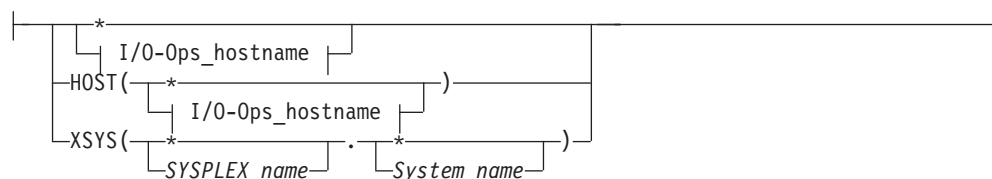
#### Range\_object\_identifier:



**Scope:**



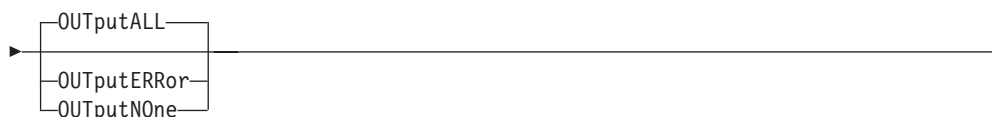
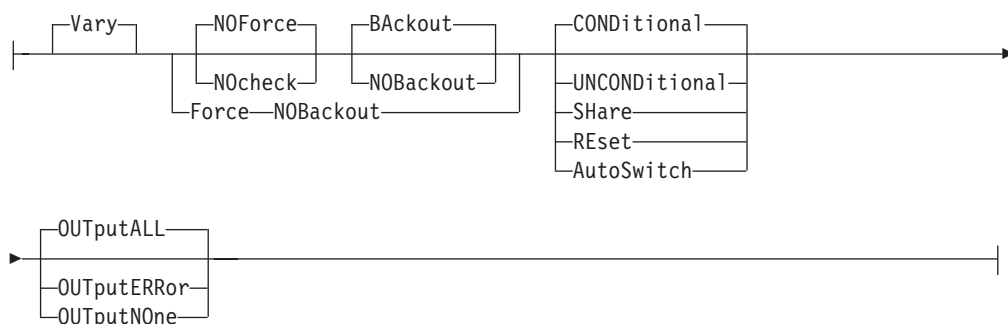
**Host\_object\_identifier:**



**I/O-Ops\_hostname:**



**Options:**



**Parameters**

A host identifier type can be one of the following keywords:

**Vary**

This is the default option and it indicates that appropriate processing must be done at the host to support the REMOVE and RESTORE DEVICE commands.

**Note:** This does not mean that the paths to this device are varied.

**NOVary**

This option is not valid for the REMOVE and RESTORE DEVICE commands.

**Force**

This option says to execute the command in the best manner possible. For example, if one of the specified hosts does not respond, the command is still performed on all other hosts.

## REMOVE DEV and RESTORE DEV

### **NOForce**

This is the default option and indicates that if there is any failure, the command should not continue and a return and reason describing the failure will be returned.

### **NOCheck**

The NOCheck option overrides the detection of two conditions that would cause the failure of the command under the default NOForce option:

1. Detection of systems in the scope of the command that I/O-Ops is not operating on
2. Detection of downlevel ESCON Managers operating on systems in the scope of the command

If either of these conditions is detected, a return code of 4 is returned.

### **BAckout**

This is the default option and indicates that if any failure is reported by any of the participating systems, any successful REMOVE and RESTORE DEVICE actions for all the participating host systems will be backed out.

### **NOBackout**

This option indicates that if any error condition is detected during the REMOVE and RESTORE DEVICE processing, I/O operations will not attempt to change any REMOVE and RESTORE DEVICE actions that have been performed.

### **CONDitional**

This is the default option for both the REMOVE and RESTORE DEVICE commands. It indicates that no special Vary offline or Vary online action should be performed.

### **UNCONDitional**

For the REMOVE DEVICE command, this option puts the specified devices immediately into pending offline status, even if the devices are currently active, allocated, or reserved.

For the RESTORE DEVICE command, this option brings the specified devices online, even if there are no paths to the devices, or if the devices are pending offline and boxed. This option is ignored if it is specified for a tape or a direct access device.

### **SHaRe**

For the REMOVE DEVICE command, this option provides no function.

For the RESTORE DEVICE command, this option permits any device that supports multisystem assign to be shared among other processors. If the device does not support multisystem assign, this option is ignored.

### **REset**

For the REMOVE DEVICE command, this option provides no function.

For the RESTORE DEVICE command, this option allows the device to be varied online even if it is currently in use by control unit initiated reconfiguration.

### **AutoSwitch**

The AutoSwitch option is valid only for a tape device such as an IBM 3480 or 3490 (or equivalent). You use Restore Dev AutoSwitch to set the option on and Remove Dev AutoSwitch to set the option off.

Setting AutoSwitch on allows a tape device to be switched serially from one system to another in a sysplex environment without the need for operator intervention.

**Note:** A coupling facility is required for sysplex tape sharing to be available.

**OUTputALL**

This is the default and it allows all results from REMOVE and RESTORE DEVICE actions that have been performed (regardless of return code) to be returned to the API invoker.

**OUTputERROR**

This option allows only error results (that is, REMOVE and RESTORE DEVICE actions with a return code > = 4, plus other errors that occurred during the processing of the command) to be returned to the API invoker.

**OUTputNone**

This option allows only the return and reason code (no text information) to be returned to the API invoker. If the return code from the command is > = 4, a detailed message (IHVC00I, IHVC001I, or IHVC002I) is also returned.

**Notes:**

1. When ARRAY is the *Object\_format\_type*, the *Object\_Identifier\_Types* can be mixed and every *Object\_Identifier\_Type* must be an I/O resource type. For example, an HNUM and an XNUM entry can be in the same array.
2. The Array\_header contains the number of elements in the array.
3. PTOK is valid with RANGE but you should be fully aware of PTOK structure. For example, RANGE PTOK could be used to specify all of the serial numbers of a certain type of device. However, certain PTOK values may cause unpredictable results with RANGE.
4. When ARRAY is the *SCOPE\_format\_type*, the *Host\_Object\_Identifier\_Types* can be mixed (HOST and XSYS).

## Output

The format of the output from the REMOVE DEV and RESTORE DEV command is as follows:

Table 21. REMOVE DEV and RESTORE DEV Output

| Offset                                 |      |              |     |             |                             |
|----------------------------------------|------|--------------|-----|-------------|-----------------------------|
| Dec                                    | Hex  | Type         | Len | Name(Dim)   | Description                 |
| 0                                      | (0)  | STRUCTURE    | *   | VDCB        |                             |
| 0                                      | (0)  | CHARACTER    | 80  | VDCB_HDR    | VDCB header                 |
| 0                                      | (0)  | CHARACTER    | 4   | VDCH_ID     | Eyecatcher ('VDCB')         |
| 4                                      | (4)  | UNSIGNED     | 2   | VDCH_HLEN   | Header length               |
| 6                                      | (6)  | UNSIGNED     | 2   | VDCH_RLEN   | Row length                  |
| 8                                      | (8)  | UNSIGNED     | 4   | VDCH_NR     | Number of rows              |
| 12                                     | (C)  | UNSIGNED     | 4   | VDCH_NHR    | Number of host summary rows |
| 16                                     | (10) | UNSIGNED     | 1   | VDCH_FMTID  | Format ID                   |
| 17                                     | (11) | CHARACTER    | 7   | *           | Reserved                    |
| Information on the command and options |      |              |     |             |                             |
| 24                                     | (18) | BITSTRING    | 2   | VDCH_CMD    | Vary device command flags   |
|                                        |      | 1... ..      |     | VDCH_VOFF   | 1 = Vary OFF device         |
|                                        |      | .1.. ..      |     | VDCH_VON    | 1 = Vary ON device          |
|                                        |      | ..1. ....    |     | VDCH_VBKOUT | 1 = Vary backout initiated  |
|                                        |      | ...1 1111 >> |     | *           | Reserved                    |

## REMOVE DEV and RESTORE DEV

Table 21. REMOVE DEV and RESTORE DEV Output (continued)

| Offset                                 |      |           |     |              |                                      |
|----------------------------------------|------|-----------|-----|--------------|--------------------------------------|
| Dec                                    | Hex  | Type      | Len | Name(Dim)    | Description                          |
| 26                                     | (1A) | BITSTRING | 2   | VDCH_OPTIONS | Vary device options flags            |
|                                        |      | 1... ..   |     | VDCH_FORCE   | 1 = Force specified                  |
|                                        |      | .1.. ..   |     | VDCH_NOFORCE | 1 = NOForce specified                |
|                                        |      | ..1. .... |     | VDCH_BKOUT   | 1 = BAckout specified                |
|                                        |      | ...1 .... |     | VDCH_NOBKOUT | 1 = NOBackout specified              |
|                                        |      | .... 1... |     | VDCH_NOCHECK | 1 = Nocheck specified                |
|                                        |      | .... .1.. |     | VDCH_COND    | 1 = CONDitional specified            |
|                                        |      | .... ..1. |     | VDCH_UNCOND  | 1 = UNConditonal specified           |
|                                        |      | .... ...1 |     | VDCH_SHARE   | 1 = SHare specified                  |
| 27                                     | (1B) | 1... ..   |     | VDCH_RESET   | 1 = REset specified                  |
|                                        |      | .1.. ..   |     | VDCH_AUTOSW  | 1 = AutoSwitch specified             |
|                                        |      | ..11 1111 |     | *            | Reserved                             |
| 28                                     | (1C) | CHARACTER | 4   | *            | Reserved                             |
| Invoker's system and user ID           |      |           |     |              |                                      |
| 32                                     | (20) | CHARACTER | 16  | VDCH_USER    |                                      |
| 32                                     | (20) | CHARACTER | 8   | VDCH_SYSID   | System ID                            |
| 40                                     | (28) | CHARACTER | 8   | VDCH_USRID   | User ID                              |
| Information on primary responding host |      |           |     |              |                                      |
| 48                                     | (30) | CHARACTER | 8   | VDCH_APPL    | I/O operations VTAM application name |
| 56                                     | (38) | CHARACTER | 16  | VDCH_SYSPLEX |                                      |
| 56                                     | (38) | CHARACTER | 8   | VDCH_SPLX    | Sysplex name                         |
| 64                                     | (40) | CHARACTER | 8   | VDCH_SYST    | System name                          |
| 72                                     | (48) | CHARACTER | 4   | VDCH_ESCMREL | SA z/OS release                      |
| 76                                     | (4C) | CHARACTER | 4   | *            | Reserved                             |
| Vary device information                |      |           |     |              |                                      |
| 80                                     | (50) | STRUCTURE | 296 | VDCB_ROW(*)  |                                      |
| 80                                     | (50) | UNSIGNED  | 2   | VDCR_FORMAT  | Row format code                      |
| 82                                     | (52) | CHARACTER | 6   | *            | Reserved                             |
| Responding host                        |      |           |     |              |                                      |
| 88                                     | (58) | CHARACTER | 8   | VDCR_APPL    | I/O operations VTAM application name |
| 96                                     | (60) | CHARACTER | 16  | VDCR_SYSPLEX |                                      |
| 96                                     | (60) | CHARACTER | 8   | VDCR_SPLX    | Sysplex name                         |
| 104                                    | (68) | CHARACTER | 8   | VDCR_SYST    | System name                          |
| Device identification                  |      |           |     |              |                                      |
| 112                                    | (70) | BITSTRING | 2   | VDCR_FLAGS   | Vary device flags                    |
|                                        |      | 1... ..   |     | VDCR_RNUMV   | 1=RNUM is valid                      |
|                                        |      | .111 1111 |     | *            | Reserved                             |
| 113                                    | (71) | 1... ..   |     | VDCR_COUPL   | 1 = Device is a coupling facility    |
|                                        |      | .1.. ..   |     | VDCR_NOVARY  | 1 = Don't vary device for row        |
|                                        |      | ..1. .... |     | VDCR_NOTFND  | 1 = Device not found for host        |
|                                        |      | ...1 .... |     | VDCR_BKOUT   | 1 = Backout attempted, msg present   |
|                                        |      | .... 1111 |     | *            | Reserved                             |
| 114                                    | (72) | UNSIGNED  | 2   | VDCR_DEVNUM  | Device number                        |
| 116                                    | (74) | BITSTRING | 4   | VDCR_SCPSTS  | Operating system state               |

Table 21. REMOVE DEV and RESTORE DEV Output (continued)

| Offset                                                                                                                                   |       |              |     |               |                                                                |
|------------------------------------------------------------------------------------------------------------------------------------------|-------|--------------|-----|---------------|----------------------------------------------------------------|
| Dec                                                                                                                                      | Hex   | Type         | Len | Name(Dim)     | Description                                                    |
| 120                                                                                                                                      | (78)  | CHARACTER    | 32  | VDCR_PTOKN    | Physical token                                                 |
| 152                                                                                                                                      | (98)  | CHARACTER    | 32  | VDCR_LTOKN    | Logical token                                                  |
| Vary results                                                                                                                             |       |              |     |               |                                                                |
| 184                                                                                                                                      | (B8)  | CHARACTER    | 96  | VDCR_VRESULTS |                                                                |
| 184                                                                                                                                      | (B8)  | BITSTRING    | 2   | VDCR_VFLAGS   | Vary flags                                                     |
|                                                                                                                                          |       | 1... ..      |     | VDCR_VMVS_MSG | 1 = Vary message is MVS                                        |
|                                                                                                                                          |       | .1.. ..      |     | VDCR_VDBCS    | 1 = Vary message is DBCS                                       |
|                                                                                                                                          |       | ..11 1111 >> | *   |               | Reserved                                                       |
| 186                                                                                                                                      | (BA)  | CHARACTER    | 2   | *             | Reserved                                                       |
| 188                                                                                                                                      | (BC)  | UNSIGNED     | 4   | VDCR_VESCMRC  | I/O operations Severity code (used for backout, msg screen)    |
| The following information is valid only when VDCR_NOVARY is not set                                                                      |       |              |     |               |                                                                |
| 192                                                                                                                                      | (C0)  | UNSIGNED     | 4   | VDCR_VMVSRC   | Return code from VARYDEV macro                                 |
| 196                                                                                                                                      | (C4)  | UNSIGNED     | 4   | VDCR_VMVSRSN  | Reason Code from VARYDEV macro                                 |
| 200                                                                                                                                      | (C8)  | CHARACTER    | 80  | VDCR_VMVSMMSG | Msg from VARYDEV macro or I/O operations based on macro RC/RSN |
| Backout results                                                                                                                          |       |              |     |               |                                                                |
| 280                                                                                                                                      | (118) | CHARACTER    | 96  | VDCR_BRESULTS |                                                                |
| 280                                                                                                                                      | (118) | BITSTRING    | 2   | VDCR_BFLAGS   | Backout flags                                                  |
|                                                                                                                                          |       | 1... ..      |     | VDCR_BMVS_MSG | 1 = Backout message is MVS                                     |
|                                                                                                                                          |       | .1.. ..      |     | VDCR_BDBCS    | 1 = Backout message is DBCS                                    |
|                                                                                                                                          |       | ..11 1111 >> | *   |               | Reserved                                                       |
| 282                                                                                                                                      | (11A) | CHARACTER    | 2   | *             | Reserved                                                       |
| 284                                                                                                                                      | (11C) | UNSIGNED     | 4   | VDCR_BESCMRC  | I/O operations Severity code                                   |
| The following information is valid only when I/O operations backout occurs (VDCH_BKOUT=1) and there are no communication errors reported |       |              |     |               |                                                                |
| 288                                                                                                                                      | (120) | UNSIGNED     | 4   | VDCR_BMVSRC   | Return code from VARYDEV macro                                 |
| 292                                                                                                                                      | (124) | UNSIGNED     | 4   | VDCR_BMVSRSN  | Reason code from VARYDEV macro                                 |
| 296                                                                                                                                      | (128) | CHARACTER    | 80  | VDCR_BMVSMMSG | Msg from VARYDEV macro or I/O operations based on macro RC/RSN |

## WRITEFILE

### Purpose

Use the WRITEFILE command at the I/O-Ops API to store a saved switch configuration at the switch specified in the command.

To use the WRITEFILE command, the switch must be allocated, or attached, to the issuing I/O-Ops.

## Syntax

▶▶—WRITEFILE—*filename*—*filedescriptor*—*datablock*—▶▶

## Parameters

### **filename**

Specify the file name in 1 through 8 valid EBCDIC codes. Valid codes are uppercase alphabetical characters (A-Z), digital characters (0-9), and 2 special characters: the underscore (\_) and the hyphen (-). However, the following file names are not valid: AUX, COM $n$  (where  $n=1-4$ ), CON, LPT $n$  (where  $n=1-3$ ), NUL, or PRN.

### **filedescriptor**

Specify the file descriptor in exactly 24 characters in the range X'40' through X'FE'.

### **datablock**

Specify a 20480-byte data block in the format listed under the Query Switch command. The data block allows an 80-byte record for 256 ports. Specify the ports in ascending hexadecimal order.

## Usage

- A maximum number of saved switch configurations can be stored at a switch. At an IBM Director, you can store up to 15 saved configurations. In addition, you can load and restore the IPL file, which is supplied with each Director and is activated automatically when the unit is powered on.
- You can only write the IPL file if the Active=Save Mode at the switch is disabled. If the mode is disabled, any changes being made to the active configuration at the switch are not saved. The mode setting is displayed on the screen of the switch console. The status is also returned in the QFILAS field in the output returned with the Q F \* command.
- You must specify the switch device number in the data block. (Unlike the WRITESWCH command, the switch device number in the Writefile command must be the same in each record.)
- If you do not want to write an entire block, you can edit an existing one. For example, you can use the Query File command to get a file, edit it, and then use the WRITEFILE command to store it.



## Input

Each 80-byte record of the WRITEFILE data block has the following format:

Table 22. WRITEFILE Input Format

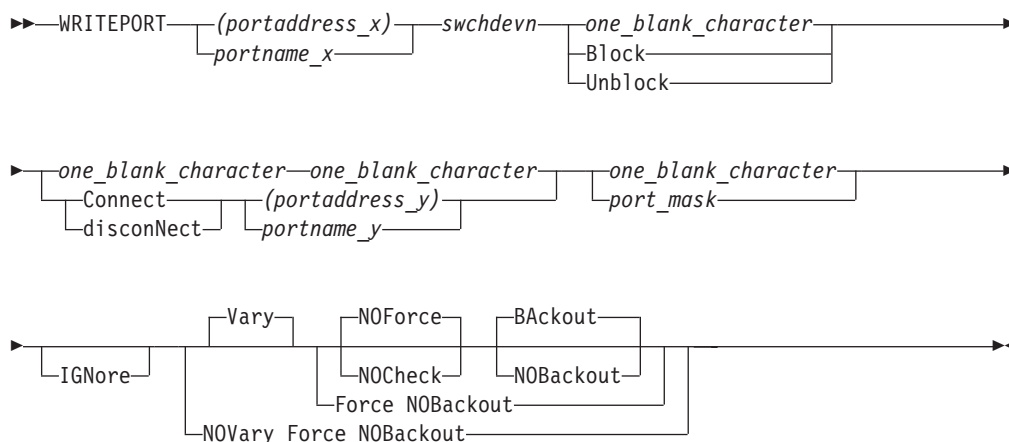
| Offset |      |              |     |           |                                                                                       |
|--------|------|--------------|-----|-----------|---------------------------------------------------------------------------------------|
| Dec    | Hex  | Type         | Len | Name(Dim) | Description                                                                           |
| 0      | (0)  | STRUCTURE    | 80  | WTFL      | Writefile record                                                                      |
| 0      | (0)  | BITSTRING    | 1   | WTFLFLAG1 | End of list indicator<br>0 = More records in list<br>1 = Last record in list          |
|        |      | 1... ..      |     | WTFLLAST  |                                                                                       |
|        |      | .1.. ....    | *   |           | Reserved                                                                              |
|        |      | ..11 .....   |     | WTFLFORM  | Format ID<br>00 = Format 0 (original format)                                          |
|        |      | .... 1111    | *   |           | Reserved                                                                              |
| 1      | (1)  | CHARACTER    | 1   | *         | Reserved                                                                              |
| 2      | (2)  | UNSIGNED     | 2   | WTFLSWIT  | Switch device number                                                                  |
| 4      | (4)  | CHARACTER    | 48  | WTFLLAIB  | Switch port information block                                                         |
| 4      | (4)  | CHARACTER    | 2   | *         | Reserved                                                                              |
| 6      | (6)  | UNSIGNED     | 1   | LAIBADDR  | Port address                                                                          |
| 7      | (7)  | CHARACTER    | 1   | *         | Reserved                                                                              |
| 8      | (8)  | BITSTRING    | 4   | LAIBDESC  | Port descriptors                                                                      |
|        |      | 1... ..      |     | LAIBUNMP  | Port implementation<br>0 = Implemented port<br>1 = Unimplemented port                 |
|        |      | .1.. ....    |     | LAIBFBIT  | Blocked port<br>0 = Port is not blocked<br>1 = Port is blocked                        |
|        |      | ..1. ....    |     | LAIBIC    | Prohibited port connection<br>0 = No prohibits for this port<br>1 = Prohibits defined |
|        |      | ...1 .....   |     | LAIBSBIT  | Port connection<br>0 = Port is not connected<br>1 = Port is connected                 |
|        |      | .... 1111 >> | *   |           | Reserved                                                                              |
| 12     | (C)  | CHARACTER    | 2   | *         | Reserved                                                                              |
| 14     | (E)  | UNSIGNED     | 1   | LAIBSADR  | Static connection address                                                             |
| 15     | (F)  | CHARACTER    | 5   | *         | Reserved                                                                              |
| 20     | (14) | BITSTRING    | 32  | LAIBICM   | Link ICM                                                                              |
| 52     | (34) | CHARACTER    | 24  | WTFLNAME  | Port logical name                                                                     |
| 76     | (4C) | CHARACTER    | 4   | *         | Reserved                                                                              |

## WRITEPORT

### Purpose

Use the Writeport command at the I/O-Ops API to define or to change the connectivity attributes for a single port on a specified switch.

## Syntax



## Parameters

### (portaddress\_x) | portname\_x

Specifies the target port by its port address (enclosed in parentheses) or by its port name.

### swchdevn | \*

Specifies the target switch device number. The switch must be allocated to, or attached to, the issuing I/O-Ops.

### one\_blank\_character | Block | Unblock

Specifies one of the following: the blocking attribute should be unchanged (X'40'); the port should be blocked; the port should be unblocked.

### one\_blank\_character | Connect | disconNect

Specifies one of the following: the dynamic connection attribute should be unchanged (X'40'); the port should be statically connected to the port specified in the next operand; the port should be disconnected from that port.

### (portaddress\_y) | portname\_y

Specifies the other port in the static connection by its port address or port name.

### one\_blank\_character | port\_mask

A blank character specifies that the allow and prohibit attributes of port\_x should be unchanged. The 256-character (32-byte) mask specifies an A (Allow) or a P (Prohibit) as the attribute for each port in the range X'00–FF'. The character representing port\_x must and all unimplemented ports must be P, while the character representing the control unit port (CUP) must be A.

### IGNore

You must specify this option when an Inter-Switch-Link port (E\_Port) is involved. Otherwise the command is rejected with return code 8 and reason code X'49'. The reason is I/O-Ops can no longer guarantee safe-switching when an E\_Port is involved.

"Safe-switching" sets the paths and devices online or offline when the path from a chpid to a device either becomes valid or is no longer valid because of a port manipulation.

**Vary**

This is the default option and it indicates that appropriate processing must be done at the host to support the REMOVE and RESTORE CHP commands.

**NOVary**

This option is not valid for the WRITEPORT command.

**Force**

This option says to do the command in the best manner possible. For example, if one of the specified hosts does not respond, the command is still performed on all other hosts.

**NOForce**

This is the default option and indicates that if there is any failure, the command should not continue and a return call and reason describing the failure will be returned.

**NOCheck**

The NOCheck option overrides the detection of two conditions that would cause the failure of the command under the default NOForce option:

1. Detection of systems in the scope of the command that I/O-Ops is not operating on
2. Detection of downlevel I/O-Ops's operating on systems in the scope of the command

If either of these conditions is detected, a return code of 4 is returned.

**BAckout**

This is the default option and indicates that if any failure is reported by any of the participating systems, any successful WRITEPORT actions for all the participating host systems will be backed out.

**NOBackout**

This option indicates that if any error condition is detected during the WRITEPORT processing, I/O-Ops will not attempt to change any WRITEPORT actions that have been performed.

## Usage

Using the Writeport command is a tool that helps you simplify the installation, set up, and recovery of a switch's configuration. Note, however, that the WRITESWCH command lets you manipulate attributes of all the ports on a switch.

The number and placement of implemented ports depends on the model of the switch. You can display the addressable ports with the commands described under "QUERY SWITCH" on page 247.

Also, see *Planning for the 9032 Enterprise Systems Connection Director* or *Planning for the 9033 Enterprise Systems Connection Director* for CUP information pertinent to the ESCON Directors.

## Examples

the following is a segment of an MVS REXX EXEC that contains the command WRITEPORT (C3) 0500 B C (C1):

```
IHVRC = 0 /* Return code; it must be */
 /* called IHVRC. */
 /* */
IHVREAS = 0 /* Reason code; it must be */
 /* called IHVREAS. */
 /* */
```

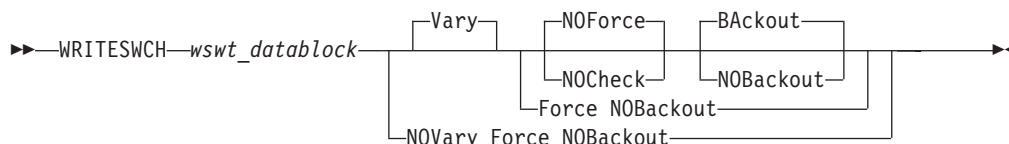




---

**WRITESWCH**
**Purpose**

Use the WRITESWCH command at the API to make changes (update) up to 512 addressable ports on any number of switches that are allocated to, or attached to, the issuing I/O-Ops. This command is available only at the API because it requires input in hexadecimal format.

**Syntax****Parameters****Vary**

This is the default option and it indicates that appropriate processing must be done at the host to support the WRITESWCH command.

**NOVary**

This option is not valid for the WRITESWCH command.

**Force**

This option says to do the command in the best manner possible. For example, if one of the specified hosts does not respond, the command is still performed on all other hosts.

**NOForce**

This is the default option and indicates that if there is any failure, the command should not continue and a return and reason describing the failure will be returned.

**NOCheck**

The NOCheck option overrides the detection of two conditions that would cause the failure of the command under the default NOForce option:

1. Detection of systems in the scope of the command that I/O-Ops is not operating on
2. Detection of downlevel I/O-Ops's operating on systems in the scope of the command

If either of these conditions is detected, a return code of 4 is returned.

**BAckout**

This is the default option and indicates that if any failure is reported by any of the participating systems, any successful WRITESWCH actions for all the participating host systems will be backed out.

**NOBackout**

This option indicates that if any error condition is detected during the WRITESWCH processing, I/O-Ops will not attempt to change any WRITESWCH actions that have been performed.

## Input

The format of the *WSWT\_datablock* is an array of 1 or more entries of the following structure:

Table 23. WRITESWCH Input

| Offset |     |           |     |           |                                                                                                              |
|--------|-----|-----------|-----|-----------|--------------------------------------------------------------------------------------------------------------|
| Dec    | Hex | Type      | Len | Name(Dim) | Description                                                                                                  |
| 0      | (0) | STRUCTURE | 80  | WSWT      |                                                                                                              |
| 0      | (0) | BITSTRING | 1   | WSWTFLAG1 | Flags byte 1                                                                                                 |
|        |     | 1... .... |     | WSWTLAST  | End of list indicator<br>0 = More records<br>1 = Last record in array                                        |
|        |     | .1.. .... |     | WSWTMDPT  | Midport<br>1 = This port is the midport<br>of a defined chain                                                |
|        |     | ..11 .... |     | WSWTFORM  | Format ID<br>0 = Format 0 (original<br>format)                                                               |
|        |     | .... 1... |     | WSWTMBSR  | Modify block state request<br>0 = No change to block state<br>1 = Change block state                         |
|        |     | .... .1.. |     | WSWTMCSR  | Modify connect state request<br>0 = No change to connect<br>state<br>1 = Change connect state                |
|        |     | .... ..1. |     | WSWTLNVB  | Logical name validity<br>0 = Ignore logical name<br>information<br>1 = Write logical name to<br>port address |
|        |     | .... ...1 |     | WSWTCIVB  | Chain information validity<br>0 = Ignore chain information<br>1 = Set up chain                               |
| 1      | (1) | BITSTRING | 1   | WSWTFLAG2 | Flags byte 2                                                                                                 |
|        |     | 1... .... |     | WSWTMMR   | Modify mask request<br>0 = No change to current<br>PDCM<br>1 = Change current PDCM                           |
|        |     | .1.. .... |     | WSWTAMR   | AND mask request<br>0 = No change to current<br>PDCM<br>1 = AND given mask with<br>current PDCM              |
|        |     | ..1. .... |     | WSWTOMR   | OR mask request<br>0 = No change to current<br>PDCM<br>1 = OR given mask with<br>current PDCM                |
|        |     | ...1 1111 |     | *         | Reserved                                                                                                     |
| 2      | (2) | UNSIGNED  | 2   | WSWTSWIT  | Switch device number                                                                                         |
| 4      | (4) | CHARACTER | 48  | WSWTLAIB  | Port information block                                                                                       |
| 4      | (4) | CHARACTER | 1   | *         | Reserved                                                                                                     |
| 5      | (5) | UNSIGNED  | 1   | LAIBNUMB  | Port number                                                                                                  |
| 6      | (6) | UNSIGNED  | 1   | LAIBADDR  | Port address                                                                                                 |
| 7      | (7) | CHARACTER | 1   | *         | Reserved                                                                                                     |
| 8      | (8) | BITSTRING | 4   | LAIBDESC  | Port descriptors                                                                                             |
|        |     | 1... .... |     | LAIBUNMP  | Port implementation<br>0 = Port is implemented<br>1 = Port is not implemented                                |
|        |     | .1.. .... |     | LAIBFBIT  | Port fence information<br>0 = Port is not blocked<br>1 = Port is blocked                                     |

Table 23. WRITESWCH Input (continued)

| Offset |      |              |     |           |                                              |  |  |
|--------|------|--------------|-----|-----------|----------------------------------------------|--|--|
| Dec    | Hex  | Type         | Len | Name(Dim) | Description                                  |  |  |
|        |      | ..1. ....    |     | *         | Reserved                                     |  |  |
|        |      | ...1 ....    |     | LAIBSBIT  | Port connection                              |  |  |
|        |      |              |     |           | 0 = Port is not connected                    |  |  |
|        |      |              |     |           | 1 = Port is connected                        |  |  |
|        |      | .... 1111 >> |     | *         | Reserved                                     |  |  |
| 12     | (C)  | CHARACTER    | 2   | *         | Reserved                                     |  |  |
| 14     | (E)  | UNSIGNED     | 1   | LAIBSADR  | Static connection port address               |  |  |
| 15     | (F)  | CHARACTER    | 5   | *         | Reserved                                     |  |  |
| 20     | (14) | BITSTRING    | 32  | LAIBICM   | Port prohibit dynamic connection mask (PDCM) |  |  |
| 52     | (34) | CHARACTER    | 24  | WSWTNAME  | Port logical name                            |  |  |
| 76     | (4C) | UNSIGNED     | 2   | WSWTCSWIT | Switch device number for chained switch      |  |  |
| 78     | (4E) | UNSIGNED     | 1   | WSWTCPORT | Chained port address                         |  |  |
| 79     | (4F) | CHARACTER    | 1   | *         | Reserved                                     |  |  |

### How to Set Up the Data Block

By using the described data block, an API user can change the connectivity attributes of a port. The changes that are requested are controlled by Request bits in the beginning of the block that must be used in order to say what type of action is requested. If no request bits are set then the given block is skipped and treated as a no-op.

The following list shows the commands that can be processed with 1 WSWT block. The bits that must be set are also listed as well as the data required to make the change.

Remember that you can make more than one change on a port block by setting the appropriate combination of bits. For example, you can effectively enter a Block and a Connect command at the same time by making sure that all the bits that are relevant for both commands are set on the same block.

#### Block

##### WSWTMSR

Must be set to 1 to indicate that the block state should be changed.

##### LAIBFBIT

Must be set to 1 to indicate that the port should be blocked.

##### LAIBADDR

Contains the port address.

##### WSWTSWIT

Contains the switch that the port address is on.

##### WSWTFORM

Must be set to 00.

#### Unblock

##### WSWTMSR

Must be set to 1 to indicate that the block state should be changed.

##### LAIBFBIT

Must be set to 0 to indicate that the port should be unblocked.



**LAIBADDR**

Contains the port address.

**WSWTSWIT**

Contains the switch that the port address is on.

**WSWTFORM**

Must be set to 00.

**Connect****WSWTMCSR**

Must be set to 1 to indicate that the connection state should be changed.

**LAIBSBIT**

Must be set to 1 to indicate that the port should be connected.

**LAIBADDR**

Contains the port address.

**LAIBSADR**

Contains the port address that LAIBADDR should be connected to.

**WSWTSWIT**

Contains the switch that the port addresses are on.

**WSWTFORM**

Must be set to 00.

**Disconnect****WSWTMCSR**

Must be set to 1 to indicate that the connection state should be changed.

**LAIBSBIT**

Must be set to 0 to indicate that the port should be disconnected.

**LAIBADDR**

Contains the port address.

**LAIBSADR**

Contains the port address that LAIBADDR should be disconnected from.

**WSWTSWIT**

Contains the switch that the port addresses are on.

**WSWTFORM**

Must be set to 00.

**Chain****WSWTCIVB**

Must be set to 1 to indicate that the chain information is valid and you want to change it.

**WSWTMDPT**

Must be set to 1 to indicate that this port is the midport on the chain.

**WSWTMCSR**

Must be set to 1 to indicate that the connection state should be changed.

**LAIBSBIT**

Must be set to 1 to indicate that the port should be chained by setting a connection between LAIBADDR and LAIBSADR.

**LAIBADDR**

Contains the port address.

## WRITESWCH

### **LAIBSADR**

Contains the port address that LAIBADDR should be connected to.

### **WSWTSWIT**

Contains the switch that the port addresses are on.

### **WSWTCPORT**

Contains the chained port address that LAIBADDR should be chained to.

### **WSWTCSWIT**

Contains the switch that the chained port address is on.

### **WSWTFORM**

Must be set to 00.

## **Unchain**

### **WSWTCIVB**

Must be set to 1 to indicate that the chain information is valid and you want to change it.

### **WSWTMDPT**

Must be set to 1 to indicate that this port is the midport on the chain.

### **WSWTMCSR**

Must be set to 1 to indicate that the connection state should be changed.

### **LAIBSBIT**

Must be set to 0 to indicate that the port should be unchained by disconnecting LAIBADDR and LAIBSADR.

### **LAIBADDR**

Contains the port address.

### **LAIBSADR**

Contains the port address that LAIBADDR should be disconnected from.

### **WSWTSWIT**

Contains the switch that the port addresses are on.

### **WSWTCPORT**

Contains the chained port address that LAIBADDR should be unchained from.

### **WSWTCSWIT**

Contains the switch that the chained port address is on.

### **WSWTFORM**

Must be set to 00.

## **Write**

### **WSWTLNVB**

Must be set to 1 to indicate that the logical name field is valid.

### **LAIBADDR**

Contains the port address.

### **WSWTNAME**

Contains the logical name that should be assigned to the port address given in LAIBADDR.

### **WSWTSWIT**

Contains the switch that the port addresses are on.

**WSWTFORM**

Must be set to 00.

**Modify PDCM****WSWTMMR**

Must be set to 1 to indicate that the PDCM should be modified.

**LAIBADDR**

Contains the port address.

**LAIBICM**

Contains the new PDCM for the given port.

**WSWTSWIT**

Contains the switch that the port addresses are on.

**WSWTFORM**

Must be set to 00.

**And PDCM****WSWTAMR**

Must be set to 1 to indicate that the given PDCM should be AND'ed with the current PDCM.

**LAIBADDR**

Contains the port address.

**LAIBICM**

Contains the PDCM to be AND'ed for the given port.

**WSWTSWIT**

Contains the switch that the port addresses are on.

**WSWTFORM**

Must be set to 00.

**Or PDCM****WSWTOMR**

Must be set to 1 to indicate that the given PDCM should be OR'ed with the current PDCM.

**LAIBADDR**

Contains the port address.

**LAIBICM**

Contains the PDCM to be OR'ed for the given port.

**WSWTSWIT**

Contains the switch that the port addresses are on.

**WSWTFORM**

Must be set to 00.

## Usage

- The WSWT data block that you enter is a series of 80-byte WSWT structures that I/O-Ops processes sequentially. Be sure to take this into account. For example, assume ports *FB* and *EA* are statically connected, and you want ports *C0* and *EA* to be statically connected instead.

## WRITESWCH

**Note:** If the WSWT structure contains an “AND PDCM” or an “OR PDCM” bit setting, there may only be one “AND PDCM”/ “OR PDCM” and no “MODIFY PDCM”, in the structure.

1. In the data block, disconnect *FB* and *EA* first.
2. Then, connect *CO*, and *EA*.

If you reverse the order, the command will fail because *EA* is already statically connected.

- If you specify Vary, I/O-Ops varies the relevant paths in all the WSWT structures offline first. If these operations are all successful or if you specified Force, the program then sends all the WSWT structures to the affected switch(es). If these operations are successful, I/O-Ops then varies the appropriate paths online.
- The Writeswch command is used implicitly when you activate a switch configuration in matrix format by using either the I/O-Ops ISPF dialog or the workstation feature.

---

## Chapter 6. Invoking I/O Operations using the API

---

### API Calls by REXX EXECs

#### Rules for Calls by a REXX EXEC

Separate the parameters as follows:

- If the REXX EXEC uses the address link invocation to call the API, the parameters must be separated by blanks as shown:  
`address link 'IHVAPI' parm1 ... parmn`
- If the REXX EXEC uses the REXX call to call the API, the parameters must be separated by commas as shown:  
`call 'IHVAPI' parm1,parm2,parm3,...`

Specify the following variables:

*ihvrc*

To receive the return code

*ihvreas*

To receive a reason code

*ihvresp*

As the response area to receive the command output if there is any

When specifying the variables listed above, note the following:

- The return code is in printable *decimal* format, while the reason code is in printable *hexadecimal* format. See *SA OS/390 Messages and Codes* for a list of reason codes.
- A return code and reason code are not provided if more parameters were specified in the input parameter list than are allowed for a REXX EXEC. A REXX error message is sent instead.
- If *ihvrc* and *ihvreas* are not specified, I/O operations can still process the command. However, the EXEC might not be able to check whether the command was processed successfully because no return code or reason code can be checked.
- If *ihvresp* is not specified, I/O operations can process the command but cannot return data to the REXX EXEC.

#### Literal Values

If the REXX EXEC calls I/O operations using literal values, the literal value for each parameter should be enclosed in single quotes (') to avoid ambiguity during processing.

#### Optional Variables

Optionally, a REXX EXEC caller can:

- Set a variable equal to the name of the command being specified
- Set a variable for each operand and option associated with the command being specified.

## Two Examples of REXX EXEC Calls

### Example of a Call to Connect Two Ports

In the following example, the caller enters the CONNECT command to connect port C0 statically, or *dedicate* it, to port E0 on switch 100.

```
/* Connect EXEC */
ihvrc = 0 /* Return code must be called */
/* */
ihvreas = 0 /* Reason code must be called */
/* ihvreas. Name must be in EXEC. */
/* */
ihvresp = ' ' /* Response area must be called */
/* ihvresp. Name must be in EXEC. */
/* */
parm1 = 'CONNECT' /* Command name */
parm2 = '(C0)' /* First port address (operand) */
parm3 = '(E0)' /* Second port address (operand) */
parm4 = '0100' /* Switch device number (operand) */
parm5 = 'NOFORCE' /* These are the default options */
parm5 = 'VARY' /* that do not have to be */
parm7 = 'BACKOUT' /* specified in the EXEC. */
```

### If the Caller Uses the Address Link Invocation:

```
address link 'IHVAPI' parm1 parm2 parm3 parm4 parm5 parm6 parm7
say "RETURN CODE = " ihvrc
say "REASON CODE = " ihvreas
say "RESPONSE AREA = " ihvresp
/* Assume screen is 80 characters - */
/* Will appear to be printing 80-character records */

EXIT
```

### If the Caller Uses the REXX Call:

```
call 'IHVAPI' parm1,parm2,parm3,parm4,parm5,parm6,parm7
say "RETURN CODE = " ihvrc
say "REASON CODE = " ihvreas
say "RESPONSE AREA = " ihvresp
/* Assume screen is 80 characters - */
/* Will appear to be printing 80-character records */

EXIT
```

## Generalized Example of a REXX EXEC Call

When processed, the following REXX EXEC can be used to enter any I/O operations command. The 80-character output is assumed to be in message format, so QUERY output will not be readable.

```
/* */
/* Initialization */
/* */
linelength = 80 /* length of 1 response line */
/* */
/* Get the command as specified by the user */
/* */
Parse Upper Arg IHVX1 IHVX2 IHVXPARMS
/* */
/* Correct basic syntax errors for the user */
/* */
/* - Capitalize command keywords (must be caps for IHV) */
/* - Strip out extraneous blanks (must have only 1 for IHV) */
/* */
IHVXCMD = Space(IHVX1 IHVX2 IHVXPARMS,1)
Drop IHVX1 IHVX2 IHVXPARMS
/* */
```

```

/* Tell the user what we are about to do */
/* */
Say 'Issuing the IHV command:' IHVXCMD */
/* */
/* Issue the command */
/* */
Address LINK 'IHVAPI' IHVXCMD */
If IHVRC > 4 */
Then /* Command failed */
Do;
 Say 'Return code:' IHVRC 'Reason code:' IHVREAS
End;
/* */
/* Show the user the response from the command */
/* */
Do lineindex = 1 to Length(IHVRESP) by lineindex;
 Say Substr(IHVRESP,lineindex,lineindex);
End;
/* */
/* Return to the caller */
/* */
Exit IHVRC;

```

---

## API Calls by the CALL Macro

### General Information

I/O operations allows a program that uses the CALL macro to invoke either IHVAPI2 or IHVAPI.

### The Parameter Lists

The caller must pass a variable-length parameter list, where:

- Each item in the list is an address of a parameter in the calling program. (The language that the program is written in must allow the program to alter the parameters for return code, reason code, and response area.)
- The high-order bit of the last parameter address must be set to 1 to indicate the end of the list.
- Register 0 must be set to 0 (zero) so that I/O operations knows the invocation is from an assembled user program, and not an interpreted REXX EXEC.
- Register 1 must contain the address of the parameter list.

### The Caller Should Check Register 15 Upon Return From the Call

If not enough parameters were passed on the CALL, I/O operations returns a reason code of X'D0xx0001' in register 15. This code specifies that either an empty or an incorrect parameter list has been sent.

- If IHVAPI2 was invoked, at least 5 parameters are needed: the command name and the last four variables listed in “The Second Parameter in the Parameter List” on page 276.
- If IHVAPI was invoked, at least 4 parameters are needed: the command name and the variables listed in “The First Parameter in the Parameter List” on page 276.

If more than 25 parameters were passed on the CALL, I/O operations returns a reason code of X'D0xx0007' in register 15. This code specifies that the list contained too many parameters.

For a comparison between IHVAPI2 and IHVAPI, refer to “Calling the I/O Operations API” on page 193.

## Calling Program Uses IHVAPI2

### Pass the Following Parameters in the Parameter List:

- One 38-character variable (padded on the right with blanks) equal to the name of the I/O operations command being specified.
- As many 38-character variables (each padded on the right with blanks) as needed for the operands in the command with the following exceptions:
  - For a range, specify a 71-character variable.
  - For an array, data block, or table, specify a variable long enough to contain it.
- As many 38-character variables (each padded on the right with blanks) as needed for the options associated with the command.
- As the fourth-from-last and the third-from-last variables, specify information related to the response area. Because these two parameters are interdependent, they are listed in the table following this list.
- As the second-from-last (or next-to-last) variable, specify a 4-byte field in hexadecimal format for the return code.
- As the last variable, specify a 4-byte field in hexadecimal format for the reason code.

| 4th-From-Last Parameter | 3rd-From-Last Parameter | When the Response Area Is To Be Managed By: |
|-------------------------|-------------------------|---------------------------------------------|
| 0 (zero)                | any value               | I/O operations with a new output buffer     |
| Response area address   | 0 (zero)                | I/O operations with a re-used output buffer |
| Response area address   | Response area length    | Caller                                      |

### Notes:

1. Initialize the response area.
2. If I/O operations manages the response area, the caller must not modify any of the fields in the prefix area, which is described in “A Prefix Area Can Precede the Response Area.” If a field is modified, the results are unpredictable.
3. On return from the call, I/O operations puts the length of the response area that it used in the third-from-last parameter. It returns a length of 0 (zero) if no response data is returned. Therefore, the caller should save the input value of this parameter before invoking I/O operations.
4. If the caller manages the response area, the caller should update the third-from-last parameter for each invocation.

### To Invoke IHVAPI2, Specify the Following:

```
CALL IHVAPI2,addrPARM1, ... addrPARMn
```

### A Prefix Area Can Precede the Response Area

If I/O operations manages the response area, it returns a prefix area as well. Use the following information when you need to release these areas.

- x        Is the address of the response area, which is contained in the fourth-from-last parameter.



- x-4 Is the 4-byte address of the prefix area, which immediately precedes the response area.
- x-12 Bytes is the 1-byte 'subpool number '0'.
- x-16 Bytes is the 4-byte length of the prefix area plus the contiguous response area.

For further information, refer to "General Information About the Response Area" on page 194 and to the following example.

### Example of a Caller Invoking IHVAPI2

```

ESCMSAMP CSECT

* Issue multisystem QUERY INTERFACE Switch to get switch port
* information. R1 points to a 4 character switch device number.

MVC SWITCH_DEVICE(4),0(R1) Get Switch device number
MVC HNUM+14(4),0(R1) Set number in query command
SR R0,R0 Required by I/O Operations
CALL IHVAPI2,(QUERY,INTERFACE,SWITCH,HNUM,VALUE, X
 ASTERISK,SCOPE,VALUE,ASTERISK, X
 QIS@,QISLENGTH,RC,REASON),VL
CLC RC,=F'0' 0 means all hosts responded ok
BNE FREE If not, then free storage

* Map the QUERY INTERFACE Switch row data.

QISOK L R10,QIS@ Point to I/O Operations output area
USING QISINFO,R10 Map query interface info
LH R9,HDRSIZE Get the QIS header size
AR R9,R10 Point to the first port row
USING PORTROW,R9 Map port interface row
:

* A port that needs blocking is found, so block it.

UNPK PORTNUMBER+1(3),PORTNUM(2) Convert 1 byte hex port
TR PORTNUMBER+1(2),TRANTAB-C'0' number to EBCDIC
MVI PORTNUMBER+3,C')' Restore trailing ")"
MVC BLKLENGTH,=F'0' Let manage the buffer
SR R0,R0 Required by I/O Operations
CALL IHVAPI2,(BLOCK,PORTNUMBER,SWITCH_DEVICE, X
 BLOCK@,BLKLENGTH,RC,REASON),VL
CLC RC,=F'4' Block worked?
BNE NOBLOCK No, then process error
:

* Now done with ESCM obtained storage, so release it.

DROP R10
FREE L R10,QIS@ Get Query output buffer
C R10,=F'0' I/O Operations Query buffer exists?
BE CONTINUE No, continue
S R10,=F'16' Address I/O Operations Query buffer
USING ESCMPREFIX,R10
L R2,BUFLNGTH Get buffer length
L R3,BUFSUBPOOL Get buffer subpool
STORAGE RELEASE,LENGTH=(R2),ADDR=BUF@,SP=(R3)
L R10,BLOCK@ Get Block output buffer
C R10,=F'0' I/O Operations Block buffer exists?
BE CONTINUE No, continue
S R10,=F'16' Address I/O Operations Query buffer

```

```

L R2,BUFLNGTH Get buffer length
L R3,BUFSUBPOOL Get buffer subpool
STORAGE RELEASE,LENGTH=(R2),ADDR=BUF@,SP=(R3)
:

* I/O Operations API parameters

QIS@ DC A(0)
QISLENGTH DC F'0'
BLOCK@ DC A(0)
BLKLENGTH DC F'0'
RC DC F'0'
REASON DC F'0'
QUERY DC CL38'QUERY'
INTERFACE DC CL38'INTERFACE'
SWITCH DC CL38'SWITCH'
VALUE DC CL38'VALUE'
ASTERISK DC CL38'*'
SCOPE DC CL38'SCOPE'
BLOCK DC CL38'BLOCK'
HNUM DC CL38'HNUM(THIS-SYS.XXXX)'
PORTNUMBER DC C'(',CL2' ',C')',CL34' '
SWITCH_DEVICE DC CL38' '
*
TRANTAB DC CL16'0123456789ABCDEF'
*
QISINFO DSECT QUERY INTERFACE Switch output
DS CL4
HDRSIZE DS H Size of this header
ROWSIZE DS H Size of each row
DS CL44
NUMROWS DS F Number of rows
*
PORTROW DSECT
PORTNUM DS XL1 Port number
DS CL155
ROWCODE DS F Query row code (see below)
PORTROW EQU 0 Port row with no error
SUMMROW EQU X'5100FFFF' Summary row
*
ESCOMPREFIX DSECT I/O Operations supplied buffer info
BUFLNGTH DS F Buffer length
BUFSUBPOOL DS FL1 Subpool number
DS F
BUF@ DS A Buffer address
:

```

## Calling Program Uses IHVAPI

### Pass the Following Parameters in the Parameter List:

- A 24-character variable (padded on the right with blanks) equal to the name of the I/O operations command being specified.
- A 24-character variable (padded on the right with blanks) for each operand in the command—with the exception of an operand that contains an array, data block, or table. In these cases, specify a variable that is long enough to contain the item. (Note, however, that I/O operations only uses 64KB of the response area on an IHVAPI call.)
- A 24-character variable (padded on the right with blanks) for each option in the command.
- As the third-from-last variable, specify the address of the response area. (Initialize the response area.)

When a caller invokes IHVAPI, I/O operations can return up to 64KB of data in the response area. If the command output exceeds this amount, I/O Operations fills the response area and notifies the caller that an overflow condition has occurred. Assume, however, that an area of 24KB is sufficient for most commands. Exceptions can be such commands as the DISPLAY DEVICE, DISPLAY RESULTS, DISPLAY VARY, QUERIES, REMOVE DEV, and RESTORE DEV commands.

- As the second-from-last variable, specify a 4-byte field in hexadecimal format for the return code.
- As the last variable, specify a 4-byte field in hexadecimal format for the reason code.

**To Call IHVAPI, Specify the Following:**

CALL IHVAPI,(CMD,PARM1,...PARMn,IHVRESP,IHVRC,IHVREAS),VL



## Part 4. Status Display Facility Definitions

|                                                           |     |                                                    |     |
|-----------------------------------------------------------|-----|----------------------------------------------------|-----|
| <b>Chapter 7. SDF Initialization Parameters</b> . . . . . | 283 | BODY . . . . .                                     | 302 |
| DCOLOR . . . . .                                          | 283 | BODYHEADER . . . . .                               | 303 |
| DPFKnn . . . . .                                          | 284 | BODYTEXT . . . . .                                 | 305 |
| DPFKDESC1 . . . . .                                       | 285 | CELL . . . . .                                     | 305 |
| DPFKDESC2 . . . . .                                       | 285 | DATETIME . . . . .                                 | 307 |
| EMPTYCOLOR . . . . .                                      | 286 | ENDPANEL . . . . .                                 | 308 |
| ERRCOLOR . . . . .                                        | 287 | INPUTFIELD . . . . .                               | 308 |
| INITSCRN . . . . .                                        | 287 | PANEL . . . . .                                    | 309 |
| MAXOPS . . . . .                                          | 288 | PFKnn . . . . .                                    | 311 |
| PFKnn . . . . .                                           | 288 | STATUSFIELD . . . . .                              | 311 |
| PRIORITY . . . . .                                        | 290 | STATUSTEXT . . . . .                               | 314 |
| PRITBSZ . . . . .                                         | 291 | TEXTFIELD . . . . .                                | 315 |
| PROPDOWN . . . . .                                        | 292 | TEXTTEXT . . . . .                                 | 317 |
| PROPUP . . . . .                                          | 292 | Example SDF Definition . . . . .                   | 318 |
| SCREENSZ . . . . .                                        | 292 | SDF Tree Structure Definitions . . . . .           | 318 |
| TEMPERR . . . . .                                         | 293 | SDF Panel Definitions . . . . .                    | 320 |
| Priority and Color Default Assignments . . . . .          | 294 | SDF Initialization Parameters in AOFINIT . . . . . | 324 |
| Subsystem . . . . .                                       | 294 | SDF Status Detail Definitions . . . . .            | 325 |
| Gateway . . . . .                                         | 295 | Example Of A Large SDF Panel . . . . .             | 325 |
| Spool . . . . .                                           | 296 |                                                    |     |
| WTOR . . . . .                                            | 296 | <b>Chapter 9. SDF Commands</b> . . . . .           | 331 |
| Monitor Resource . . . . .                                | 296 | Using SDF Commands . . . . .                       | 331 |
| Captured Message . . . . .                                | 296 | Dynamically Loading Panels and Tree Structures     | 331 |
| TWS Automation . . . . .                                  | 297 | SCREEN . . . . .                                   | 333 |
| Groups . . . . .                                          | 297 | SDF . . . . .                                      | 333 |
| Processor Operations . . . . .                            | 297 | SDFCONF . . . . .                                  | 334 |
|                                                           |     | SDFPANEL . . . . .                                 | 335 |
| <b>Chapter 8. SDF Definition Statements</b> . . . . .     | 299 | SDFTREE . . . . .                                  | 336 |
| AOFTREE . . . . .                                         | 299 | Navigation Commands . . . . .                      | 337 |

This part describes the definitions for the status display facility (SDF). See *IBM Tivoli System Automation for z/OS User's Guide* for information about how to set up the display panels and how to use SDF.



---

## Chapter 7. SDF Initialization Parameters

The SDF initialization parameters are:

|                   |                                                                                               |
|-------------------|-----------------------------------------------------------------------------------------------|
| <b>DCOLOR</b>     | Default status descriptor color, see “DCOLOR”                                                 |
| <b>DPFKnn</b>     | PF key settings for detail status panel, see “DPFKnn” on page 284                             |
| <b>DPFKDESC1</b>  | PF key descriptions for detail status panel, see “DPFKDESC1” on page 285                      |
| <b>DPFKDESC2</b>  | PF key descriptions for detail status panel, see “DPFKDESC2” on page 285                      |
| <b>EMPTYCOLOR</b> | Default color for status component without a status descriptor, see “EMPTYCOLOR” on page 286  |
| <b>ERRCOLOR</b>   | Default color for status component without a tree structure entry, see “ERRCOLOR” on page 287 |
| <b>INITSCRN</b>   | Initial screen, see “INITSCRN” on page 287                                                    |
| <b>MAXOPS</b>     | Maximum operator logon limit, see “MAXOPS” on page 288                                        |
| <b>PFKnn</b>      | Default PF key settings, see “PFKnn” on page 288                                              |
| <b>PRIORITY</b>   | Priority and color definitions, see “PRIORITY” on page 290                                    |
| <b>PRITBLSZ</b>   | Priority and color table size, see “PRITBLSZ” on page 291                                     |
| <b>PROPDOWN</b>   | Propagate status downward in SDF tree structure, see “PROPDOWN” on page 292                   |
| <b>PROPUP</b>     | Propagate status upward in SDF tree structure, see “PROPUP” on page 292                       |
| <b>SCREENSZ</b>   | Screen size, see “SCREENSZ” on page 292                                                       |
| <b>TEMPERR</b>    | Temporary error limit value, see “TEMPERR” on page 293                                        |

---

### DCOLOR

#### Purpose

The DCOLOR parameter defines the color that is used for a status descriptor that is outside any of the defined priority and color ranges. This parameter is optional. If it is not coded, the program default color is White.

#### Syntax



#### Parameters

*color*

The color that is used for the status descriptor. It can be one of the following:

**R** Red

## DCOLOR

**P** Pink  
**Y** Yellow  
**T** Turquoise  
**G** Green  
**B** Blue  
**W** White

The default is White.

### Restrictions and Limitations

In member AOFINIT, if the number of PRIORITY parameters (see "PRIORITY" on page 290) exceeds the default PRITBLSZ parameter value of 7 (see "PRITBLSZ" on page 291), the DCOLOR parameter must follow the PRITBLSZ parameter.

### Usage

The recommended value for DCOLOR is White. It is supplied in the SA z/OS SINGNPRM member AOFINIT. It does not conflict with existing status and color definitions.

### Examples

```
DCOLOR = WHITE
```

---

## DPFKnn

### Purpose

The DPFK $nn$  parameter defines all PF keys unique to a detailed status panel.

### Syntax

▶▶—DPFK $nn$ =*command*—▶▶

### Parameters

*nn* PF key number. Values can range from 1 to 24. You can modify all PF key definitions.

*command*

The command executed when the defined PF key is pressed.

### Restrictions and Limitations

This parameter must be specified on one line. Continuation lines are not allowed. The total length of the parameter and parameter value specification cannot exceed 72 characters.

PF keys defined by DPFK $nn$  statements are only active when the detail panel is displayed and override the default settings defined with the PFK $nn$  parameter.

### Usage

Table 24 on page 289 shows variables that you can use as part of the command specified on the DPFK $nn$  parameter.



## Examples

DPFK9 = SCREEN VTAMSTAT

### DPFKDESC1

#### Purpose

The DPFKDESC1 parameter defines the first part of the PF key description appearing at the bottom of the detail screen. This text is concatenated with the text defined with the DPFKDESC2 parameter.

#### Syntax

▶▶—DPFKDESC1=*text*—————▶▶

#### Parameters

*text*

The text of the detail PF key description. The length of text allowed for this parameter depends on the total parameter length limit (72 characters) and the total text length limit defined by DPFKDESC1 and DPFKDESC2 (80 characters). For example, when defining a detail PF key description that is 79 characters long, you can define the first 60 characters of text on DPFKDESC1 and the remainder of the text on DPFKDESC2.

#### Restrictions and Limitations

- This parameter must be specified on one line. Continuation lines are not allowed. The total length of the parameter and parameter value specification cannot exceed 72 characters.
- The total length of the PF key description defined by DPFKDESC1 and DPFKDESC2 cannot exceed 80 characters.

#### Examples

DPFKDESC1=PF3=RET      6=ROLL 7=UP 8=DN 9=AST 10=DEL

### DPFKDESC2

#### Purpose

The DPFKDESC2 parameter defines the second part of the PF key description appearing at the bottom of the detail screen. This text is concatenated with the text defined with the DPFKDESC1 parameter.

#### Syntax

▶▶—DPFKDESC2=*text*—————▶▶

#### Parameters

*text*

The text of the continued PF key description, begun in a previous DPFKDESC1

statement. The length of the text depends on the length specified on the previous DPFKDESC1 statement, because the total description text defined by DPFKDESC1 and DPFKDESC2 cannot exceed 80 characters.

## Restrictions and Limitations

- This parameter must be specified on one line. Continuation lines are not allowed. The total length of the parameter and parameter value specification cannot exceed 72 characters.
- The total length of the PF key description defined by DPFKDESC1 and DPFKDESC2 cannot exceed 80 characters.

## Examples

```
DPFKDESC2=11=BOT 12=TOP
```

---

## EMPTYCOLOR

### Purpose

The EMPTYCOLOR parameter defines the color displayed for a status component that has no status descriptor associated with it. This parameter is optional. If it is not coded, the default color is Blue.

### Syntax



### Parameters

#### *color*

The color that is used for the status descriptor. It can be one of the following:

**R** Red  
**P** Pink  
**Y** Yellow  
**T** Turquoise  
**G** Green  
**B** Blue  
**W** White

The default is Blue.

### Usage

The recommended value for EMPTYCOLOR is Blue. It is supplied in SA z/OS SAOFNPRM member AOFINIT. It does not conflict with existing status and color definitions. This parameter can be overridden in the AOFTREE member.

### Examples

```
EMPTYCOLOR = BLUE
```

---

## ERRCOLOR

### Purpose

The ERRCOLOR parameter defines the color displayed for a status component that does not have a corresponding entry in the SDF tree structure.

This parameter is optional. If it is not coded, the default color is White.

### Syntax



### Parameters

#### *color*

The color that is used for the status component. It can be one of the following:

- R** Red
- P** Pink
- Y** Yellow
- T** Turquoise
- G** Green
- B** Blue
- W** White

The default is White.

### Examples

```
ERRCOLOR = YELLOW
```

---

## INITSCRN

### Purpose

The INITSCRN parameter defines the initial panel displayed by SDF.

### Syntax



### Parameters

#### *panel\_name*

Any valid alphanumeric name with a maximum length of sixteen.

### Usage

If you change the name of the initial panel defined in the AOFPNLS member of the NetView DSIPARM data set, you must also change the panel name in the INITSCRN parameter.

## Examples

```
INITSCRN = SYSTEMA1
```

## MAXOPS

### Purpose

The MAXOPS parameter defines the maximum number of logged-on operators that can use the SDF. This parameter is optional. If it is not coded, a program default of 30 is used.

### Syntax



### Parameters

*number*

The number of maximum operators. Values can range from 1 to 999. The default is 30.

### Usage

If the number of operators trying to use the SDF is more than the number defined in MAXOPS, additional operators are denied access to the SDF, because the dynamic update facility keeps an internal count of logged-on operators.

### Examples

```
MAXOPS = 35
```

## PFKnn

### Purpose

The PFKnn parameter defines the default PF key settings for SDF panels.

### Syntax



### Parameters

*nn* Values can range from 1 to 24.

*command*

The command issued when the defined PF key is pressed.

### Restrictions and Limitations

This parameter must be specified on one line. Continuation lines are not allowed. The total length of the parameter and parameter value specification cannot exceed 72 characters.

## Usage

Table 24 shows the variables that can be used as part of the command specified on the PFKnn parameter.

### Notes:

1. Use of these variables (that is, their appropriate translation from variables to values) is valid on a detail status panel or on a status panel when the cursor is on a status field.
2. The '#' character can be used alternatively instead of the '&' character in order to avoid name clashes with system symbol names.

Table 24. Variables for the PFKnn and DPFKnn Parameters

| Variable                     | Translated To                                                                     |
|------------------------------|-----------------------------------------------------------------------------------|
| &CO or &COLOR                | The color of the detail entry                                                     |
| &COMP or &RESOURCE           | The component name                                                                |
| &DA or &DATA or<br>&DISPDETL | The actual message text                                                           |
| &DATE                        | The date the detail entry was added                                               |
| &DCOMP                       | The displayed component name                                                      |
| &HL or &HIGHLIGHT            | The highlight level of the detail entry                                           |
| &IN or &INFO                 | Detail entry information displayed on the status panel                            |
| &PR or &PRIORITY             | The priority of the detail entry                                                  |
| &QCOMP                       | The component name that the status was queued to by SDF                           |
| &RESAPPL or &COMPAPPL        | The component name and the alternate component name, if used, to queue the status |
| &ROOT or &SYSTEM             | Root or system                                                                    |
| &RV or &REFVALUE             | The reference value of the detail entry                                           |
| &SENDERID or &SID            | The reporter submitting the detail entry                                          |
| &SNODE or<br>&SENDERNODE     | The node of the reporter submitting the detail entry                              |
| &SO or &SOURCE               | The name of the reporter submitting the detail entry                              |
| &SYSDATE                     | System date                                                                       |
| &SYSTEMTIME                  | System time                                                                       |
| &TIME                        | The time the detail entry was added                                               |

## Examples

To issue MVS D A,TS0 when PF4 is pressed with the cursor placed on the TSO entry on the status screen:

```
PFK4 =MVS D A,&INFO
```

The following example assigns the SDFCONF command to the PF4 key to delete SDF entries. This is useful because it prompts you for confirmation before performing the actual deletion. If you do not want the prompt panel to appear, then add ",VERIFY=NO" to the end of the SDFCONF command.

**Note:** VFY can be used as an abbreviation for VERIFY should a parameter length restriction apply.

```
PFK4=SDFCONF &ROOT,&COMPAPPL,&RV,&SID,&SNODE,&DATE,&TIME,&DA
```

| When SDFCONF deletes an exceptional message entry from the SDF control  
 | structure it also removes the message from all other interfaces that the message has  
 | been sent to, for example, TEP and NMC.

## PRIORITY

### Purpose

The PRIORITY parameter defines the relationship between colors and priority ranges. This parameter is optional. If it is not coded, program defaults are used.

### Syntax



### Parameters

*nnn*

The lower limit of the priority range. It can be any valid number between 001 and 99999999.

*mmm*

The upper limit of the priority range. It can be any valid number between 001 and 99999999. It must be equal to or greater than the value specified in *nnn*.

*color*

The color that is used for a particular priority range. It can be one of the following:

- R** Red
- P** Pink
- Y** Yellow
- T** Turquoise
- G** Green
- B** Blue
- W** White

*program\_defaults*

These are:

| Priority Range | Color     |
|----------------|-----------|
| 001–199        | RED       |
| 200–299        | PINK      |
| 300–399        | YELLOW    |
| 400–499        | TURQUOISE |
| 500–599        | GREEN     |
| 600–699        | BLUE      |

### Restrictions and Limitations

- This parameter must be specified on one line. Continuation lines are not allowed. The total length of the parameter and parameter value specification cannot exceed 72 characters.

- In the AOFINIT member, if the number of PRIORITY parameters defining priority and color ranges (see “PRIORITY” on page 290) exceeds the default PRITBLSZ parameter value of 7 (see “PRITBLSZ”), the DCOLOR parameter must follow the PRITBLSZ parameter.
- Default values for priorities and colors are used if and only if no PRIORITY parameters are defined. If you choose to customize any priority and color definitions, you must specify all priority and color definitions in AOFINIT, rather than customizing the one priority and color definition and using the defaults for the remaining definitions.

## Usage

It is recommended that you use the priority and color values that are supplied with the SA z/OS DSIPARM member AOFINIT.

## Examples

```
Priority = 001,199,RED
Priority = 200,299,PINK
Priority = 300,399,YELLOW
Priority = 400,499,TURQUOISE
Priority = 500,599,GREEN
Priority = 600,699,BLUE
```

## PRITBLSZ

### Purpose

The PRITBLSZ parameter defines the number of priority and color ranges defined by the PRIORITY entries. This parameter is optional. The default is 7.

### Syntax



### Parameters

*nn* The number of priority and color ranges. It can be any number greater than or equal to 7. The default is 7.

### Restrictions and Limitations

In the AOFINIT member, if the number of PRIORITY parameters defining priority and color ranges (see “PRIORITY” on page 290) exceeds the default PRITBLSZ parameter value of 7 (see “PRITBLSZ”), the DCOLOR parameter must follow the PRITBLSZ parameter.

### Usage

The recommended value for PRITBLSZ is 7. It is supplied with the SA z/OS DSIPARM member AOFINIT.

### Examples

```
PRITBLSZ = 7
```

## PROPDOWN

---

### PROPDOWN

#### Purpose

The PROPDOWN parameter defines whether status information should be sent down the status tree as a system default or not. This parameter is optional. The default is NO.

#### Syntax



#### Parameters

None.

#### Usage

The recommended value for PROPDOWN is NO. This parameter can be overridden with individual requests to add a status descriptor to a status component.

---

## PROPUP

#### Purpose

The PROPUP parameter defines whether status information should be sent up the status tree as a system default. This parameter is optional. The default is YES.

#### Syntax



#### Parameters

None.

#### Usage

The recommended value for PROPUP is YES. This parameter can be overridden with individual requests to add a status descriptor to a status component.

---

## SCREENSZ

#### Purpose

The parameter SCREENSZ defines the buffer size as well as the screen size. The screen size is used for verifying panel definitions of panels with an undefined screen size.



The first value of the SCREENSZ parameter defines the screen buffer size. The buffer size must be as large as the largest panel size. The size of each panel can be calculated as follows:

- Count the number of CELL definitions that fit into the associated body section. Repeat this for each body section
- Accumulate the counts
- Multiply the sum by 17
- Add the length of each CELL definition that has been counted in step 1
- Count the STATUSFIELD, TEXTFIELD, and BODYHEADER definitions
- Multiply the count by 13 and add the value to the result of step 4
- Add the length of each STATUSFIELD, TEXTFIELD, and BODY
- Add twice the maximum panel width
- Add 32
- Round the total size to the next higher multiple of 8

### Syntax



### Parameters

*number*

Buffer size value. Values can range from 3000 to 99999. The default is 3000.

*rows*

The minimum number of rows. Values can range from 24 to 62. The default is 24.

*cols*

The minimum number of columns. Values can range from 80 to 160. The default is 80.

### Examples

SCREENSZ = 4000,43,80

## TEMPERR

### Purpose

The TEMPERR parameter defines the maximum number of temporary input/output errors when trying to display a SDF panel. This parameter is optional. The default is 3.

### Syntax



## TEMPERR

### Parameters

*number*

Values can range from 3 to 99. The default is 3.

### Usage

The recommended value for TEMPERR is 3. It is supplied with the SA z/OS DSIPARM member AOFINIT.

### Examples

TEMPERR = 3

---

## Priority and Color Default Assignments

In SDF and the DISPSTAT command dialog, subsystems appear in different colors that indicate their status. The condition of WTORs, application groups (APGs), monitor resources (MTRs), tapes requests, TWS automation messages and gateways are also indicated by color.

You can modify the default assignments when you define your SA z/OS policy. Refer to the "Status Display Entry Type" in *IBM Tivoli System Automation for z/OS Defining Automation Policy* for how to change the default assignments or to define your own status values.

Some of the default assignments show the value 'NOADD' in the field 'Req'. This means that the component with the corresponding status is not added to the SDF chain.

**Note:** Combining NOADD with 'Clear=Y' causes the component to be removed from the SDF display when this status is reached.

## Subsystem

The color and highlighting of a subsystem in SDF indicate the current status of a subsystem (the possible status conditions are defined in "Automation Agent Statuses" in *IBM Tivoli System Automation for z/OS User's Guide*).

Table 25 shows the default color and highlighting that is assigned to each status by SA z/OS. White is also used by default to identify status components without a tree structure. These colors may have been changed for your SDF display.

Blinking as a means of highlighting is not used by every type of display, in particular when you are using 3270 terminal emulation on a PC. You may see another type of highlighting, for example, black text on a white background.

*Table 25. Table of Default Subsystem Status Colors and Priorities*

| Status   | Priority | Color | Highlight |
|----------|----------|-------|-----------|
| BROKEN   | 120      | Red   | Reverse   |
| BREAKING | 130      |       | Blinking  |
| STOPPED  | 140      |       | Normal    |

## Priority and Color Default Assignments

Table 25. Table of Default Subsystem Status Colors and Priorities (continued)

| Status   | Priority | Color     | Highlight |
|----------|----------|-----------|-----------|
| HALFDOWN | 220      | Pink      | Normal    |
| STARTED2 | 230      |           | Blinking  |
| STUCK    | 240      |           | Underline |
| ZOMBIE   | 250      |           | Reverse   |
| ABENDING | 320      | White     | Reverse   |
| HALTED   | 330      |           | Underline |
| STOPPING | 420      | Yellow    | Reverse   |
| AUTOTERM | 430      |           | Normal    |
| ENDING   | 440      |           | Underline |
| RUNNING  | 520      | Turquoise | Blinking  |
| ACTIVE   | 530      |           | Reverse   |
| STARTED  | 540      |           | Underline |
| RESTART  | 550      |           | Normal    |
| EXTSTART | 550      |           | Blinking  |
| UP       | 640      | Green     | Normal    |
| ENDED    | 650      |           | Underline |
| DOWN     | 730      | Blue      | Underline |
| AUTODOWN | 740      |           | Normal    |
| INACTIVE | 740      |           | Normal    |
| CTLDOWN  | 750      |           | Reverse   |
| MOVED    | 760      |           | Reverse   |
| FALLBACK | 770      |           | Normal    |

Each automation status is assigned a priority number that SDF uses to determine what color to display if there are multiple status conditions present in a system. The status conditions are listed in Table 25 on page 294 in order of descending priority, that is, statuses listed first have higher priorities, and their color is displayed.

## Gateway

In addition to displaying the status of subsystems, SDF uses colors to show the status of gateway sessions. The following list explains what the gateway colors indicate:

Table 26. Table of Default Gateway Status Colors and Priorities

| Status   | Priority | Color     | Highlight | Clear |
|----------|----------|-----------|-----------|-------|
| COMMLOST | 110      | Red       | Reverse   | Y     |
| ACTIVE   | 530      | Turquoise | Reverse   | Y,RV* |

## Priority and Color Default Assignments

### Spool

SDF uses the following colors to show the status of spools.

Table 27. Table of Default Spool Status Colors and Priorities

| Status   | Priority | Color  | Highlight | Clear | Req   |
|----------|----------|--------|-----------|-------|-------|
| SPLFULL  | 150      | Red    | Normal    | Y,RV  | –     |
| SPLSHORT | 450      | Yellow | Normal    | Y,RV  | –     |
| SPLOK    | 550      | Green  | Normal    | Y     | –     |
| SPLGONE  | 550      |        | –         | Y,RV  | NOADD |

### WTOR

SDF also uses colors to show what type of WTOR each displayed WTOR is, based on classifications that have been specified using the customization dialog. See *IBM Tivoli System Automation for z/OS Defining Automation Policy* for more details.

Table 28. Table of Default WTOR Status Colors and Priorities

| Status | Priority | Color  | Highlight | Clear | Req   |
|--------|----------|--------|-----------|-------|-------|
| CWTOR  | 50       | Red    | Normal    | Y     | –     |
| IWTOR  | 250      | Pink   | Normal    | Y     | –     |
| UWTOR  | 450      | Yellow | Normal    | Y     | –     |
| NWTOR  | 550      | Green  | Normal    | Y     | –     |
| RWTOR  | 750      | Green  | –         | Y,RV  | NOADD |

### Monitor Resource

SDF uses the following colors to show the status of monitor resources (MTRs).

Table 29. Table of Default Monitor Resource Status Colors and Priorities

| Status   | Priority | Color     | Highlight | Clear |
|----------|----------|-----------|-----------|-------|
| FATAL    | 110      | Red       | Reverse   | Y,RV* |
| CRITICAL | 140      |           | Normal    | Y,RV* |
| MINOR    | 410      | Yellow    | Blinking  | Y,RV* |
| WARNING  | 450      |           | Normal    | Y,RV* |
| FAILED   | 550      | Turquoise | Normal    | Y,RV* |
| NORMAL   | 650      | Green     | Normal    | Y,RV* |
| UNKNOWN  | 730      | Blue      | Normal    | Y,RV* |

### Captured Message

Captured exceptional messages that are shown in SDF have the following status definitions:

Table 30. Table of Default Captured Message Status Colors and Priorities

| Status | Priority | Color  | Highlight | Clear | Req   |
|--------|----------|--------|-----------|-------|-------|
| CMSG   | 50       | Red    | Normal    | Y     | –     |
| IMSG   | 250      | Pink   | Normal    | Y     | –     |
| UMSG   | 450      | Yellow | Normal    | Y     | –     |
| DELETE | 750      | Green  | –         | Y     | NOADD |

## TWS Automation

TWS automation messages that are shown in SDF have the following status definitions:

Table 31. Table of Default TWS Automation Status Colors and Priorities

| Status  | Priority | Color | Highlight | Clear | Req   |
|---------|----------|-------|-----------|-------|-------|
| OPCERR  | 190      | Red   | Normal    | Y     | –     |
| BTCHEND | 750      | Green | –         | Y,RV* | NOADD |

## Groups

Groups that are shown in SDF have the following status definitions:

Table 32. Table of Default Group Status Colors and Priorities

| Status   | Priority | Color  | Highlight | Clear |
|----------|----------|--------|-----------|-------|
| APG_PROB | 150      | Red    | Normal    | Y,RV* |
| APG_ERR  | 450      | Yellow | Normal    | Y,RV* |
| APG_OK   | 650      | Green  | Normal    | Y,RV* |
| APG_DOWN | 750      | Blue   | Normal    | Y,RV* |

## Processor Operations

Processor Operations (ProcOps) elements that are shown in SDF have the following status definitions.

**Note:** To avoid any conflict with other status elements each ProcOps status element begins with the # (hash) sign.

Table 33. Table of Ensemble Status Colors and Priorities

| Status   | Priority | Color     | Highlight | ProcOps Ensemble Status |
|----------|----------|-----------|-----------|-------------------------|
| #SESSPRB | 140      | Red       | Normal    | SESSION PROBLEM         |
| #SUSPEND | 250      | Pink      | Normal    | SUSPENDED               |
| #DORMANT | 350      | White     | Normal    | DORMANT                 |
| #CONNECT | 380      | White     | Normal    | CONNECTING              |
| #CLOSED  | 450      | Turquoise | Normal    | CLOSED                  |
| #ACTIVE  | 550      | Green     | Normal    | ACTIVE                  |
| #UNKNOWN | 680      | Blue      | Normal    | UNKNOWN                 |

## Priority and Color Default Assignments

Table 34. Table of Processor Status Colors and Priorities

| Status   | Priority | Color     | Highlight | ProcOps Processor Status |
|----------|----------|-----------|-----------|--------------------------|
| #PROBLEM | 120      | Red       | Reverse   | LPAR DEFINITION PROBLEM  |
| #PROBLEM | 120      | Red       | Reverse   | TARGET HARDWARE PROBLEM  |
| #SERVREQ | 220      | Pink      | Normal    | SERVICE REQUIRED         |
| #SUSPEND | 250      | Pink      | Normal    | SUSPENDED                |
| #PORREQ  | 320      | Yellow    | Normal    | POWER-ON-RESET REQUIRED  |
| #SERVICE | 330      | Yellow    | Normal    | SERVICE                  |
| #DEGRAD  | 340      | Yellow    | Normal    | DEGRADED                 |
| #POWSAV  | 420      | Turquoise | Normal    | POWER <sup>®</sup> SAVE  |
| #ACTIVE  | 550      | Green     | Normal    | OK                       |
| #ACTIVE  | 550      | Green     | Normal    | OPERATING                |
| #POWOFF  | 600      | Blue      | Normal    | POWERED OFF              |
| #UNKNOWN | 680      | Blue      | Normal    | UNKNOWN                  |

Table 35. Table of LPAR Status Colors and Priorities

| Status   | Priority | Color     | Highlight | ProcOps LPAR Status |
|----------|----------|-----------|-----------|---------------------|
| #DISWAIT | 110      | Red       | Blink     | DISABLED WAIT       |
| #CRIALRT | 150      | Red       | Normal    | SERIOUS ALERT       |
| #LOADERR | 160      | Red       | Normal    | LOAD FAILED         |
| #DCCF    | 170      | Red       | Normal    | DCCF                |
| #NOTOPER | 240      | Pink      | Normal    | NOT OPERATING       |
| #SUSPEND | 250      | Pink      | Normal    | SUSPENDED           |
| #CLOSED  | 450      | Turquoise | Normal    | CLOSED              |
| #INITIAL | 540      | Green     | Normal    | INITIALIZED         |
| #IPLCOMP | 550      | Green     | Normal    | IPL COMPLETE        |
| #INACTIV | 650      | Blue      | Normal    | NOT ACTIVE          |
| #UNKNOWN | 680      | Blue      | Normal    | UNDECIDABLE         |
| #UNKNOWN | 680      | Blue      | Normal    | UNKNOWN             |

---

## Chapter 8. SDF Definition Statements

The status display facility (SDF) provides a display of automated systems and resources using assigned status colors. An operator monitors the status of automated systems and resources by viewing the SDF main panel.

Typically, an application shown in green on an SDF status panel indicates the application is up, while red indicates the application is stopped or in a problem state. Operators can use the SDF to monitor the system and decide what actions to take when problems occur.

See *IBM Tivoli System Automation for z/OS Defining Automation Policy* for information on how to define the SDF in the customization dialogs. You only need to change these entries if you use values other than the SA z/OS-provided defaults.

**Attention:** Using a single quote as a parameter requires you to specify it as four single quotes. Furthermore, when a single quote is part of a string it must be replaced by two single quotes. Or, when used as a string delimiter, the string or each partial string, if the string is spread over two or more lines, must be enclosed in single quotes. Otherwise, unpredictable results occur.

---

### AOFTREE

#### Purpose

AOFTREE is a NetView DSIPARM member containing tree structure definitions, or referencing other tree structure definition members by using %INCLUDE statements. The tree structure definitions specify the propagation hierarchy that is used for status color changes.

#### Syntax

Each tree structure definition entry must be in the following format:

►► *level\_number*—*status\_component* —————►  
                                          └ *empty\_chain\_color* ─┘

#### Parameters

##### *level\_number*

The level number assigned to each component in the tree structure. It can be any valid number between 1 and 9999. A tree structure must start with the root as level number 1.

If a level number is less than that of the preceding status component, the level number that is used must be defined in the tree structure as a superior node to that status component. For example, the following tree structure definition is *incorrect*:

```
1 SY1
3 APPLIX
2 GATEWAY
```

Multiple roots can be defined in the same member, using 1 as the level number.

*status\_component*

The status component that is associated with the *level\_number*. It can be any application or subsystem that status information is to be displayed for. System symbols are supported for the status component name. This can help reduce both customization work and errors.

Uses the subsystem entry name as defined in the automation control file. The status component entry for the root must match the SDFROOT value specified on the SA z/OS Environment Setup panel in the customization dialogs that define the current automation policy.

*empty\_chain\_color*

The color that a status component is displayed in on the SDF status panel if no status descriptor is associated with a status component. It can be one of the following:

- R Red
- P Pink
- Y Yellow
- T Turquoise
- G Green
- B Blue
- W White

This entry is optional. If it is not coded, the value specified for the SDF initialization parameter EMPTYCOLOR in member AOFINIT is used. See "EMPTYCOLOR" on page 286 for more details.

## Usage

SA z/OS uses the following status components:

|  |                                                       |                                 |
|--|-------------------------------------------------------|---------------------------------|
|  |                                                       |                                 |
|  | <b>BATCH</b>                                          | Batch job related events        |
|  | <b>CPMSGS</b>                                         | Exceptional messages            |
|  | <b>GATEWAY</b>                                        | Gateway events                  |
|  | <b>GROUPS</b>                                         | APG resources                   |
|  | <b>MONITOR</b>                                        | MTR resources                   |
|  | <b>OPCERR</b>                                         | TWS events                      |
|  | <b>SPOOL</b>                                          | Spool events                    |
|  | <b>SUBSYS</b>                                         | APL resources (subsystems)      |
|  | <b>TAPE</b>                                           | Tape-related events             |
|  | <b>TSOUSERS</b>                                       | TSO users                       |
|  | <b>WTOR</b>                                           | WTOR messages                   |
|  |                                                       |                                 |
|  | <b>Processor Operations related status components</b> |                                 |
|  | <b>ENS</b>                                            | zEnterprise® ensemble resources |
|  | <b>TGT</b>                                            | LPAR resources                  |
|  | <b>THW</b>                                            | Processor resources             |

When creating tree structure definitions, consider the following:

- Level numbers define the order of dependence. As an example, in "Examples" on page 301, AOFAPPL is defined to depend on AOFSSI because AOFAPPL relies on AOFSSI for its message traffic. With propagation, any AOFSSI status change is reflected on both AOFAPPL and SY1 status components.
- Duplicate status components in the same tree structure should not be used.



- Not all status components defined in a tree structure require a corresponding panel entry. That is, you can define entries in a tree structure that do not have a corresponding panel display. However, every panel should have a corresponding entry in the tree structure.
- To avoid addressing conflicts, each root name must be unique. SDF addresses each status component defined in the tree structure as `root_component.status_component`

## Examples

This example defines two separate tree structures, SY1 and SY2, representing two different MVS systems. SY1 is the focal point and SY2 is the target system.

Figure 15 and Figure 16 on page 302 show the tree structures that must be defined in the tree structure definition member for SY1. /\* denotes a comment field.

```

/* TREE STRUCTURE FOR SYSTEM SY1
1 SY1
2 SUBSYS
3 AOFAPPL
4 AOFSSI
3 JES2
4 SPOOL
3 VTAM
3 RMF

```

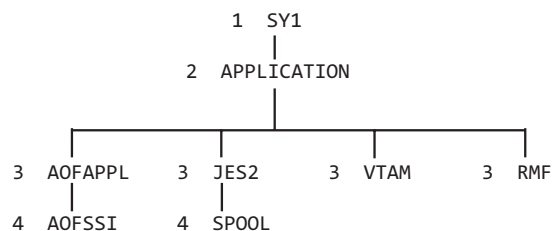


Figure 15. Example Tree Structure Definitions: System SY1. The diagram following the tree structure code for SY1 shows how the order of dependence relates to level number. The diagram is not actually in AOFTREE.

```

/* TREE STRUCTURE FOR SYSTEM SY2 ON SY1
1 SY2
2 SUBSYS
3 AOFAPPL
4 AOFSSI
3 JES2
4 SPOOL
3 VTAM
3 RMF
3 TSO

```

## AOFTREE

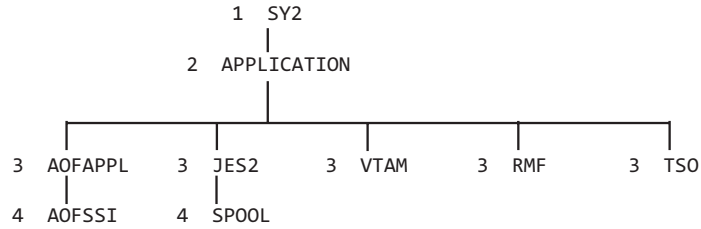


Figure 16. Example Tree Structure Definitions: System SY2. The diagram following the tree structure code for SY1 and SY2 shows how the order of dependence relates to level number. The diagram is not actually in AOFTREE.

These tree structures are referenced in the AOFTREE member on SY1 by the following %INCLUDE statements:

```
%INCLUDE(SY1TREE)
%INCLUDE(SY2TREE)
```

The AOFTREE member in system SY2 contains only a %INCLUDE statement referencing the tree structure for SY2.

Both tree structures start with level number 1. While the tree structures have unique root names, they can have similar status component names, such as JES2, and TSO. The corresponding settings for the root component can be defined in the system policy and SYSTEM INFO definitions.

---

## BODY

### Purpose

The BODY statement is used to define the section in the panel that can be used by SDF to display the various status components listed in the order of their priority, as well as the layout of the table.

### Syntax

```
►► BODY (status_comp, start_line, end_line, [1] #_cols, [5] distance)
►, [1] start_pos, [*] end_pos)
```

### Parameters

#### *status\_comp*

The name of the status component as defined in the AOFTREE member. It can be optionally prefixed with the root component name.

#### *start\_line*

The line number where the body section begins. You can specify either the absolute number or use relative addressing based on the bottom line of the panel. Relative addressing uses a notation if \*-n where n is the displacement from the bottom line.

#### *end\_line*

The last line of the body section. You can specify either the absolute number or

use relative addressing based on the bottom line of the panel. Relative addressing uses a notation of \*-n where *n* is the displacement from the bottom line.

#### *#\_cols*

Specifies the number of columns to be generated. The default is 1.

This variable can also be asterisk (\*). In this case, SA z/OS attempts to fill the entire BODY width with columns.

#### *distance*

Specifies the distance between columns. The default is 5.

#### *start\_pos*

Specifies the column number where the body section begins. The default is 1.

#### *end\_pos*

Specifies the column number where the body section ends. The default is the screen width.

## Restrictions and Limitations

The distance between two columns appears one character more than as specified. This is caused by a 'START FIELD' order which terminates each cell.

You can specify up to 16 BODY sections per panel.

## Examples

The examples below assume a 43x80 screen size. The next example shows the definition that is needed for a Subsystem Status display where the panel body starts at line 4 and runs to line 39. The section begins at column 1 and ends at column 80. The panel layout is 2 columns.

```
BODY(&SDFROOT..SUBSYS,04,*-4,2,2)
```

This example shows the definition that is needed for a Monitor Status display where the panel body starts at line 23 and runs to line 40. The section begins at column 41 and ends at column 80. The panel layout is as many columns as fit in 40-character width.

```
BODY(&SDFROOT.MONITOR,*-20,*-3,*1,41,*)
```

---

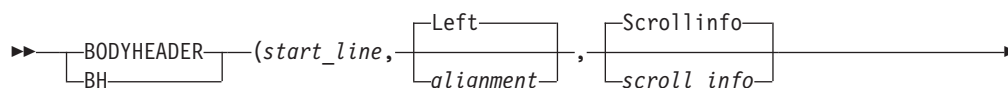
## BODYHEADER

### Purpose

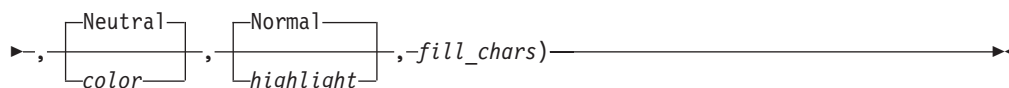
The BODYHEADER statement is used to define a descriptive header and the scrolling information of the body section. The length of the header is implicitly defined by the body width.

**Note:** The scrolling information is only displayed when the header is long enough to show the complete information, that is, "bnn/nnn(nnn)". It may overwrite the title defined by BODYTEXT.

### Syntax



## BODYHEADER



### Parameters

#### *start\_line*

The line number that the header should be displayed on. You can specify either the absolute number or use relative addressing based on the bottom line of the panel. Relative addressing uses a notation of \*-n where *n* is the displacement from the bottom line.

#### *alignment*

The justification of the text specified in the corresponding BODYTEXT statement. It can be one of the following:

- C** Center
- L** Left (default)

#### *scroll\_info*

Specifies whether scroll information is displayed or not. It can be one of the following:

- S** Scroll information is displayed right-justified at the end of the header line (default).
- N** No scroll information is displayed.

#### *color*

The color that the text specified in the corresponding BODYTEXT statement is displayed in. It can be one of the following:

- B** Blue
- G** Green
- N** Neutral (=White)
- P** Pink
- R** Red
- T** Turquoise
- W** White
- Y** Yellow

#### *highlight*

Specifies the type of highlighting to be used for information units of the status component. The value can be one of the following:

- N** Normal (default)
- B** Blink
- R** Reverse
- U** Underscore

#### *fill\_chars*

You can specify a string of up to 6 characters that replaces the spaces in the header line.

### Restrictions and Limitations

Not more than one header can be specified per body.

This statement must follow its associated 'BODY' definition statement.

## Examples

This example defines the header line being on line 3. The title is centered and supplemented with scroll information on the right side. Spaces are replaced by hyphens. The whole text is displayed in turquoise, and uses reverse highlighting.

```
BH(03,Center,,Turquoise,Reverse,-)
```

## BODYTEXT

### Purpose

The BODYTEXT statement defines the text displayed in the panel header line (see “BODYHEADER” on page 303).

### Syntax

```

▶▶ BODYTEXT (text) ◀◀
 └── BT ───┘

```

### Parameters

*text*  
The text displayed in the header line. The length of the text determines the width of the corresponding BODY definition.

### Restrictions and Limitations

The total length of the BODYTEXT cannot be exceed the field length defined by the combination of BODY *start\_pos* and *end\_pos* parameter values.

This statement must follow its associated 'BODY' definition statement.

### Usage

To continue a BODYTEXT statement, insert a delimiting comma and leave the remaining columns up to and including column 72 blank. Resume the text definition in column 1 of the following line. See “TEXTTEXT” on page 317 for an example of a continued statement.

## Examples

The example shows the definitions that are needed for the Subsystem Status display from “BODY” on page 302.

```
BT(Subsystem Status)
```

## CELL

### Purpose

The CELL statement is used to define the various information units to be displayed for a status component and their placement in the body section.

### Syntax

## CELL

▶▶—CELL—(*start\_pos*—,—*end\_pos*—,—*highlight*—,—*X*)—▶▶

### Parameters

#### *start\_pos*

Specifies the starting position that the status component information unit is to be placed on.

#### *end\_pos*

Specifies the position where the information unit ends. You can also specify an asterisk (\*) to indicate that the information unit ends at the end of the body section.

#### *highlight*

Specifies the type of highlighting to be used for this information unit of the status component. The value can be one of the following:

- N** Normal
- B** Blink
- R** Reverse
- U** Underscore

The highlight attribute is overwritten by the corresponding attribute of the status component descriptor when specified. Refer to “Priority and Color Default Assignments” on page 294 for the actual highlighting value.

**X** Denotes the type of information to be displayed. If omitted, the MVS job name is displayed if the resource is a subsystem or WTOR.

See “STATUSFIELD” on page 311 for list of valid types.

To allow for the attribute type, there must be a minimum of two spaces between the ending position of one field and the beginning position of the next. For example, if the end-position of a CELL is in column 10, the start-position of the next CELL must be column 12 or later.

### Restrictions and Limitations

- You can specify a maximum of 12 cells.
- A *start\_pos* parameter value of 1 is not allowed.

### Examples

This example shows the definitions that are needed for the Subsystem Status display from “BODY” on page 302. The gap between the columns is 2 plus the indentation of the first cell, which is 5. There are three information units shown in each column line. The information units are

- Job name
- Date
- Time

```
CELL(05,12,N)
CELL(14,21,N,D)
CELL(23,30,N,T)
```

## DATETIME

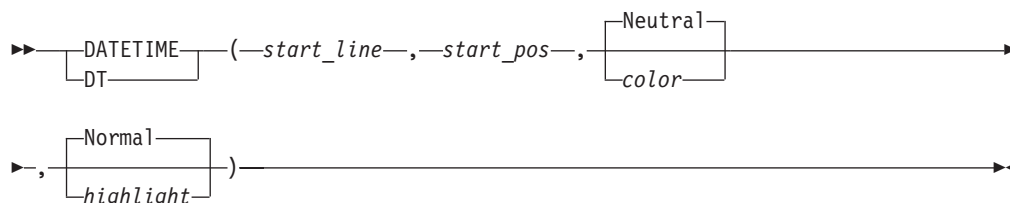
### Purpose

The DATETIME statement defines the location and attributes of the current date and time. When specified, it replaces the date and time displayed on the message line.

The field has a fixed length of 17. The presentation of the date and time values derives from the NetView DEFAULTS templates LONGDATE and LONGTIME.

### Syntax

Parameters are positional.



### Parameters

#### *start\_line*

The line number that the field should be displayed on. You can specify either the absolute number or use relative addressing based on the bottom line of the panel. Relative addressing uses a notation of \*-n where *n* is the displacement from the bottom line.

The resulting value must be in the range specified in the *length* parameter in the PANEL definition statement (see "PANEL" on page 309).

#### *start\_pos*

The column number that the field is placed in. You can specify either the absolute number or use relative addressing based on the width of the panel. Relative addressing uses a notation of \*-n where *n* is the displacement from the rightmost column.

The resulting value must be in the range specified in the *width* parameter in the PANEL definition statement (see "PANEL" on page 309) minus the length of this field.

#### *color*

The color that the date and time are displayed in. It can be one of the following:

- R** Red
- P** Pink
- Y** Yellow
- T** Turquoise
- G** Green
- B** Blue
- W** White (=Neutral)

#### *highlight*

Determines how the date and time are displayed. It can be one of the following:

## DATETIME

|                    **N** Normal  
|                    **B** Blink  
|                    **R** Reverse  
|                    **U** Underscore

### Restrictions and Limitations

- The length of the field is fixed to 17 characters.
- Not more than one DATETIME field can be specified per panel.

### Examples

This example defines the DATETIME field as being on line 1, positioned at the end of the line. The date and time are displayed in white, and use normal highlighting.

```
DATETIME(01,*-16,WHITE,NORMAL)
```

---

## ENDPANEL

### Purpose

The ENDPANEL statement identifies the end of a panel.

### Syntax

The diagram shows the syntax for the ENDPANEL statement. It starts with a double arrow pointing right. Below the arrow, the text "ENDPANEL" is enclosed in a box. To its right, the text "EP" is enclosed in a box. Further right, the text "(panel\_name)" is enclosed in a box. A long horizontal line extends from the end of the "(panel\_name)" box to the right, ending in a double arrow pointing right.

### Parameters

*panel\_name*

The name of the panel. This parameter is optional. If specified, this parameter value must match the name specified on the previous PANEL statement.

### Restrictions and Limitations

None.

---

## INPUTFIELD

### Purpose

The INPUTFIELD (IF) statement defines the location of the input field. Previously, the input field was the penultimate line of the panel. The IF statement gives you the flexibility to place the input field (command line) anywhere in the panel. In addition, it lets you define the message line to be placed below or above the input field.

### Syntax

The diagram shows the syntax for the INPUTFIELD statement. It starts with a double arrow pointing right. Below the arrow, the text "INPUTFIELD" is enclosed in a box. To its right, the text "(start\_line,—,—start\_pos,—," is enclosed in a box. Further right, the text "Above" is enclosed in a box above the text "msg\_pos", which is enclosed in a box below. A long horizontal line extends from the end of the "msg\_pos" box to the right, ending in a double arrow pointing right.



## Parameters

### *start\_line*

The line number that the input field should be displayed on. You can specify either the absolute number or use relative addressing based on the bottom line of the panel. Relative addressing uses a notation of \*-*n* where *n* is the displacement from the bottom line.

The resulting value must be in the range specified in the *length* parameter in the PANEL definition statement (see "PANEL").

### *start\_pos*

The offset in the specified line where the input fields begins.

### *msg\_pos*

The position of the message line relative to the input field. The value can be one of the following:

**A** Above (default)

**B** Below

The input field is automatically terminated by any BODY, STATUSFIELD, or TEXTFIELD statement placed after it, either on the same line or the next.

## Restrictions and Limitations

Not more than one input field can be specified per panel.

## Examples

This example shows the definition that is needed for an input field that is 2 lines above the last line:

```
IF(*-2,01)
```

---

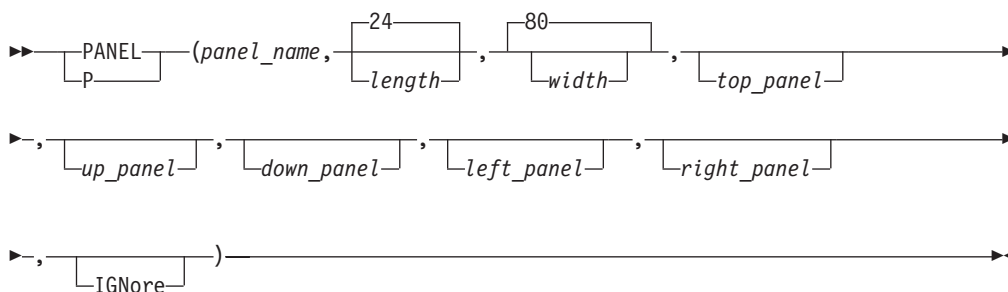
## PANEL

### Purpose

The PANEL statement identifies the start of a new panel and its general attributes.

### Syntax

Parameters are positional.



## Parameters

*panel\_name*

The name of the panel. It can be any panel name up to 16 characters long.

*length*

The number of lines or rows in the panel. It must be numeric. Supported values are 24, 32, 43, 62, or \* (the screen size at logon). The default is 24.

*width*

The number of columns in the panel. It must be numeric. This can be 80, 132, 160, or \* (the screen width at logon). The default is 80.

*top\_panel*

The panel displayed when the TOP PF key is pressed or the TOP command is issued.

*up\_panel*

The panel displayed when the UP PF key is pressed or the UP command is issued.

*down\_panel*

The panel displayed when the DOWN PF key is pressed or the DOWN command is issued.

*left\_panel*

The panel displayed when the left panel PF Key is pressed or the LEFT command is issued. The panel name can be \*, allowing vertical scrolling through the displayed information of a body section.

*right\_panel*

The panel displayed when the right panel PF key is pressed or the RIGHT command is issued. The panel name can be \*, allowing vertical scrolling through the displayed information of a body section.

*IGNore*

This option causes SDF to ignore the UP/DOWN command when used in a BODY section.

## Usage

- The default initial panel name that is supplied with SA z/OS is SYSTEM. If you change this name, also change the INITSCRN parameter value in the AOFINIT member (see "INITSCRN" on page 287 for details).
- If there is more data than can be displayed on a single screen, you can define continuation panels using the following parameters:
  - *left\_panel*
  - *right\_panel*
  - *down\_panel*
- To continue a PANEL statement on another line after a delimiting comma, leave the remaining columns up to and including column 72 blank. The next positional parameter must begin in column 1 of the following line.

## Examples

This example defines SY1SYS as the panel name. The length is 24 lines and the width is 80 characters. The panel named SYSTEM is displayed when the TOP and UP commands are used. No entries are defined for the DOWN, LEFT, or RIGHT commands.

```
PANEL(SY1SYS,24,80,SYSTEM,SYSTEM)
```

---

## PFKnn

### Purpose

The PFKnn entry defines all PF keys unique to a panel.

The definitions defined by PFKnn are only active when the status panel is displayed. They override the default settings defined with the initialization PFKnn statement in member AOFINIT (see “PFKnn” on page 288).

### Syntax

```
►► PFKnn (command, [variable])
```

### Parameters

*nn* The PF key number. It can range from 1 through 24.

#### *command*

The command called when the defined PF key is pressed. If you need to use commas in the command, enclose the entire command string in apostrophes.

#### *variable*

Variables can be used as part of the command specified in the PFKnn statement. “PFKnn” on page 288 shows variables that can be used.

### Examples

This example results in issuing MVS D A,TSO when PF4 is pressed and the cursor is on the TSO entry: PFK4('MVS D A,&INFO')

---

## STATUSFIELD

### Purpose

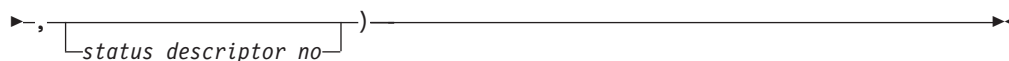
The STATUSFIELD statement defines the location of the status component on a panel and the panels that display when the UP and DOWN commands are used.

A STATUSFIELD statement is always accompanied by a STATUSTEXT statement (see “STATUSTEXT” on page 314) in a panel definition.

### Syntax

Parameters are positional.

```
►► STATUSFIELD (root_comp.status_comp, —start_line—, —start_pos—
SF
, —end_pos—, [Normal]
[highlight], [up_panel], [down_panel])
```



## Parameters

### *root\_component*

The root component name as defined in the root node of the tree structure. The root component (as opposed to the status component alone) must always be coded, because different systems can have status components with the same name, such as VTAM or JES2, in their tree structures. Because the root component is always unique, each status component in a tree structure can be uniquely identified using the root component as a prefix.

### *status\_comp*

The status component name as defined in the AOFTREE member. Maximum length is 16 characters.

### *start\_line*

The line number that the status component should be displayed on. You can specify either the absolute number (it should be numeric and in the range specified in the *length* parameter of the PANEL definition statement) or you can use relative addressing based on the bottom line of the panel. Relative addressing uses a notation of \*-n where n is the displacement from the bottom line.

### *start\_pos*

The actual column number within the specified *start\_line* that the status component is to be placed on. There must be a minimum of two spaces between the ending position of one field and the beginning position of the next field to allow for attribute type. For example, if the end-position of a STATUSFIELD is in column 10, the start-position of the next STATUSFIELD must be column 12.

### *end\_pos*

The column number that the status component definition ends in. It is governed by the length of text defined in the STATUSTEXT definition. For example, if JES2 is to be defined, the length of the STATUSTEXT is four and the end position is the start position plus three. See "STATUSTEXT" on page 314 for more details.

### *highlight*

The type of highlighting that is used on the panel. It can be one of the following:

- N** Normal
- B** Blink
- R** Reverse
- U** Underscore

The highlight attribute is overwritten by the corresponding attribute of the status component descriptor when specified. Refer to "Priority and Color Default Assignments" on page 294 for the actual highlighting value.

### *up\_panel*

The panel displayed when the UP PF key is pressed.

### *down\_panel*

The panel displayed when the DOWN PF key is pressed.

*status\_descriptor\_number*

The status descriptor number of the panel. This number specifies the status descriptor displayed in each field. It must be numeric. The default is 0.

A status descriptor number of 0 causes the text as defined in the STATUSTEXT statement for this field (see “STATUSTEXT text Parameter” on page 314) to be displayed with the color and highlighting associated with the first status descriptor chained to the status component. A status descriptor of 1 essentially does the same, except that the status text is replaced by information contained in the first status descriptor chained to the status component. A status descriptor of 2 or higher has the same effect as a value of 1, except that the numbered status descriptor is used rather than the first.

Status descriptors are chained with the status component in ascending order of priority.

The status descriptor number may be prefixed with a letter denoting the type of information to be displayed. If no prefix is supplied, the MVS job name is displayed if the resource is a subsystem or WTOR. Valid prefixes are as follows:

- C** Displays the name of the status component
- D** Displays the date the record was created
- M** Displays the associated message text
- P** Displays the priority of the record
- Q** Displays the reference value for the record
- R** Displays the name of the root component
- S** Displays the reporting operator ID
- T** Displays the time the record was created
- U** Displays the number of duplicate records
- V** Displays the job name or other information about the request
- X** Displays the reporting domain ID

## Restrictions and Limitations

A *start\_line* and *start\_position* parameter value combination of 1,1 is not allowed.

SDF panels that contain STATUSFIELD entries that refer to status descriptors other than the first one may not be updated dynamically if the panel is not made resident either using the SDFPANEL ...,ADD command or during SDF initialization. Automatic updates on dynamically loaded panels may be obtained by coding a dummy panel containing STATUSFIELD entries that refer to the status component with a status descriptor number greater than 1.

At least one undefined (blank) position must be provided immediately preceding a STATUSFIELD. If *start\_position* is specified as 1, the last position on the preceding line must not be defined.

## Usage

- When designing a panel for any status component, make the end position greater than or equal to the start position. Otherwise, an error condition will occur during SDF initialization.
- To continue a STATUSFIELD statement on another line after a delimiting comma, leave the remaining columns up to and including column 72 blank. The next positional parameter must begin in column 1 of the following line. An example of a continued STATUSFIELD statement is:

```
STATUSFIELD(SY.VTAM,
04,10,13,NORMAL)
```

## STATUSFIELD

- For better performance, make sure that every status component referred to in the panel is defined in the corresponding AOFTREE member.

### Examples

#### Example 1

In this example, the status component VTAM on SY1 starts on line 4 in column 10, ends in column 13, and has normal highlighting. No entries are defined for the UP or DOWN commands.

```
STATUSFIELD(SY1.VTAM,04,10,13,NORMAL)
:
```

#### Example 2

In this example, the status component SYSTEM starts on line 2 in column 04, ends in column 06, and has normal highlighting. No entries are defined for the UP panel. Panel SY1SYS is displayed when the DOWN command is issued.

```
SF(SY1.SYSTEM,02,04,06,N,,SY1SYS)
:
```

#### Example 3

In this example, three STATUSFIELD entries are defined for the same status component, SY1.GATEWAY. The highest-priority status descriptor is displayed in the first entry, the next highest-priority status descriptor is displayed in the second entry, and so on.

```
SF(SY1.GATEWAY,02,04,06,NORMAL,,1)
SF(SY1.GATEWAY,03,04,06,NORMAL,,2)
SF(SY1.GATEWAY,04,04,06,NORMAL,,3)
:
```

---

## STATUSTEXT

### Purpose

The STATUSTEXT statement defines the text data displayed in the STATUSFIELD statement (see “STATUSFIELD” on page 311). This text data is typically the status component name.

### Syntax

```
ST—(text)—————
```

### Parameters

#### *text*

The default data displayed for the status component defined in the STATUSFIELD statement. This text can be replaced by text from a status descriptor chained to the status component if the *status\_descriptor\_number* parameter value on the corresponding STATUSFIELD statement is non-zero. The recommended value is the status component name. For example, for status component SY1.VTAM, specify VTAM for the *text* value. The length of text determines the end position coded in the STATUSFIELD statement.

## Restrictions and Limitations

- Each STATUSFIELD statement must have a STATUSTEXT statement associated with it in a panel definition.
- The total length of the STATUSTEXT text cannot exceed the status field length defined by the combination of STATUSFIELD *start\_position* and *end\_position* parameter values.
- This statement must follow its associated 'STATUSFIELD' definition statement.

## Usage

To continue a STATUSTEXT statement, insert a delimiting comma and leave the remaining columns up to and including column 72 blank. Resume the text definition in column 1 of the following line.

## Examples

### Example 1

The following statement defines status text 1234567890 for a status field:

```
STATUSTEXT(12345,
67890)
```

### Example 2

This example defines that IMS on SY2 displays as ACCOUNTS on the status display panel. Any status descriptors added for SY2.IMS are displayed using the ACCOUNTS entry.

**Note:** The end position in the STATUSFIELD statement reflects the length of ACCOUNTS.

```
STATUSFIELD(SY2.IMS,06,10,17,NORMAL)
STATUSTEXT(ACCOUNTS)
```

---

## TEXTFIELD

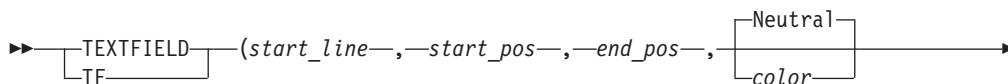
### Purpose

The TEXTFIELD statement defines the location and attributes of fields that remain constant on the panels, such as panel headings, field names, and PF key designations.

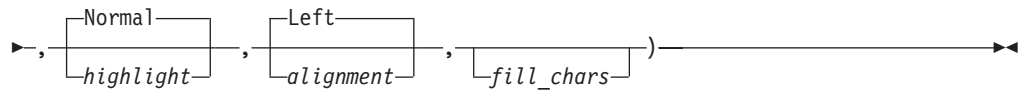
Each TEXTFIELD statement must have a TEXTTEXT statement associated with it (see "TEXTTEXT" on page 317) in a panel definition.

### Syntax

Parameters are positional.



## TEXTFIELD



### Parameters

#### *start\_line*

The line number that the text field should be displayed on. You can specify either the absolute number or use relative addressing based on the bottom line of the panel. Relative addressing uses a notation of \*-*n* where *n* is the displacement from the bottom line.

The resulting value must be in the range specified in the *length* parameter in the PANEL definition statement (see "PANEL" on page 309).

#### *start\_pos*

The column number that the text field is placed in.

#### *end\_pos*

The column number that the data specified in entry TEXTTEXT ends in. See "TEXTTEXT" on page 317 for more details.

#### *color*

The color that text specified in the corresponding TEXTTEXT statement is displayed in. It can be one of the following:

- R** Red
- P** Pink
- Y** Yellow
- T** Turquoise
- G** Green
- B** Blue
- W** White (=Neutral)

#### *highlight*

Determines how the text specified in the corresponding TEXTTEXT statement is displayed. It can be one of the following:

- N** Normal (default)
- B** Blink
- R** Reverse
- U** Underscore

#### *alignment*

The alignment of the text specified in the corresponding TEXTTEXT statement. It can be one of the following:

- C** Center
- L** Left (default)

#### *fill\_chars*

You can specify a string of up to 6 characters that replaces the spaces in the text field.

### Restrictions and Limitations

- A *start\_line* and *start\_position* parameter value combination of 1,1 is not allowed.
- If your text definition for an area of a panel requires more than 72 characters, continue the definition in additional TEXTFIELD and TEXTTEXT statement pairs. See "TEXTTEXT" on page 317 for an example of continuing definitions in additional TEXTFIELD and TEXTTEXT pairs.



- At least two undefined (blank) positions must be provided immediately preceding a TEXTFIELD if it follows a STATUSFIELD. If *start\_position* is specified as 1, the last two positions on the preceding line must not be defined.
- At least one undefined (blank) position must be provided immediately preceding a TEXTFIELD if it follows another TEXTFIELD. If *start\_position* is specified as 1, the last position on the preceding line must not be defined.

## Usage

- When designing a panel, for any TEXTFIELD, make the *end\_position* of the TEXTFIELD greater than or equal to the *start\_position*. Otherwise, an error condition will occur during SDF initialization.
- To continue a TEXTFIELD statement on another line after a delimiting comma, leave the remaining columns up to and including column 72 blank. The next positional parameter must begin in column 1 of the following line. An example continued TEXTTEXT statement is:

```
TEXTFIELD(01,
25,57,WHITE,NORMAL)
```

## Examples

This example defines the TEXTFIELD as being on line 1, starting in column 25, ending in column 57. The text is displayed in white, and uses normal highlighting.

```
TEXTFIELD(01,25,57,WHITE,NORMAL)
```

## TEXTTEXT

### Purpose

The TEXTTEXT statement defines the data displayed in the corresponding TEXTFIELD entry (see “TEXTFIELD” on page 315).

Each TEXTFIELD statement must have a TEXTTEXT statement associated with it in a panel definition.

### Syntax

```

▶▶—TEXTTEXT—(text)—————▶▶
 |_____|
 TT

```

### Parameters

*text*

The data displayed for the TEXTFIELD statement. The length of the data determines the end position coded in the TEXTFIELD entry.

### Restrictions and Limitations

The total length of the TEXTTEXT text cannot exceed the text field length defined by the combination of TEXTFIELD *start\_position* and *end\_position* parameter values.

This statement must follow its associated 'TEXTFIELD' definition statement.

|

## TEXTTEXT

### Usage

To continue a TEXTTEXT statement, insert a delimiting comma and leave the remaining columns up to and including column 72 blank. Resume the text definition in column 1 of the following line. See the “TEXTTEXT” on page 317 for an example continued statement.

### Examples

#### Example 1

In this example, “Data center systems” is displayed on the status display panel in white.

```
| TEXTFIELD(01,25,57,WHITE,NORMAL)
| TEXTTEXT(Data center systems)
```

#### Example 2

In this example, all PF key settings are displayed on line 24 of the status display panel.

```
TF(24,01,79,TURQUOISE,NORMAL)
TEXTTEXT(PF1=HELP 2=DETAIL 3=END 6=ROLL 7=UP 8=DN ,
10=LF 11=RT 12=TOP)
```

---

## Example SDF Definition

This section shows an example of defining SDF. This example is not reflected by the SA z/OS DSIPARM member AOF TREE. In this example, two separate systems (SY1 and SY2) are defined to SDF so that SDF can monitor both systems. The example shows the entries that are required to define and customize SDF, including:

- SDF tree structure definitions
- SDF panel definitions
- SDF initialization parameters in AOFINIT
- SDF Status Details definitions

#### Notes:

1. This example assumes that SA z/OS focal point services are already implemented so that status can be forwarded from one system to another using notification messages.
2. It is also assumed that the advanced automation option (AAO) common global AOF\_AAO\_SDFROOT\_LISTn defines the SDF root names that are to be applied (see *IBM Tivoli System Automation for z/OS Customizing and Programming*). If the common global is not specified SA z/OS will automatically set the global to the local system name.

### SDF Tree Structure Definitions

Two tree structure definitions are required to set up the SDF hierarchy for systems SY1 and SY2. Figure 17 on page 319 shows the tree structure definition for both systems. This tree structure is defined in a NetView DSIPARM data set member named SY1TREE.

```

1 &SDFROOT.
2 SYSTEM
3 JES
3 VTAM
3 RMF
3 TSO
3 AOFAPPL
4 AOFSSI
3 APPLIC
4 SUBSYS
2 ACTION,GREEN
3 WTOR,GREEN
2 GATEWAY

```

Figure 17. SDF Example: Tree Structure Definition for SY1 and SY2

SDF initialization uses the common global AOF\_AA0\_SDFROOT\_LISTn to generate as many tree structures as system names are found in the global for each structure that starts with 1 &SDFROOT..

Figure 18 shows the hierarchy of monitored resources defined by the SY1 tree structure.

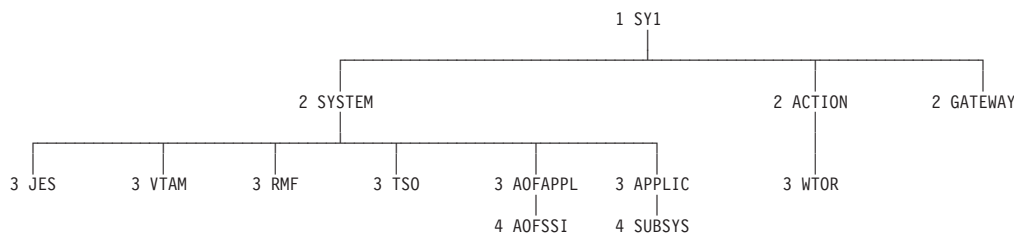


Figure 18. SDF Example: Hierarchy Defined by SY1 Tree Structure. The diagram shows how the order of dependence relates to level number.

This structure contains specific entries for the major system components, JES, RMF™, VTAM, and TSO, as well as NetView (AOFAPPL) and the NetView SSI (AOFSSI). Note that the hierarchy differs from that defined in the SA z/OS automation control file. This is because the operator's view of these subsystems differs from the logical sequence that they are managed in by SA z/OS for startup and shutdown purposes.

The SYSTEM, APPLIC, and ACTION entries are logical, and may be used to view the status of all entries below them in priority order.

The SUBSYS, WTOR, and GATEWAY entries are also logical, and may be used to display the status of SUBSYSTEM, WTOR, and GATEWAY resource types. The status of any subsystem not appearing elsewhere in the tree will be queued under the SUBSYS entry. Similarly, WTORS and gateway status will be queued under WTOR and GATEWAY respectively.

A similar tree structure must be provided for SY2. As both systems are running the same set of base software, the tree structures are identical, except for the root (level 1) name, which will be SY2 rather than SY1. The member holding the tree structure must be defined in the NetView DSIPARM data set on each system. The tree structure is referenced by %INCLUDE statements in the base SDF tree definition member, AOFTREE, as follows:

```
%INCLUDE(SYXTREE)
```

## SDF Definitions

| Because SY1 is the focal point system in this example, the global  
| AOF\_AA0\_SDFROOT\_LISTn on system SY1 must define both systems, SY1 and SY2.

| Because our example does not require SY2 to function as a backup focal point  
| system, the global AOF\_AA0\_SDFROOT\_LISTn system SY2 does not need to be  
| specified.

## SDF Panel Definitions

Following are panel definitions for:

- The root component or system panel, named SYSTEM
- The status component panel for system SY1, named SY1

Each panel definition is followed by the screen it defines.

SA z/OS provides samples similar to those described, as well as a sample GATEWAY panel definition for use on SY1.

On system SY1, these panel definitions are referenced by %INCLUDE statements in the main SDF panel definition member, AOFPNLS, as follows:

The GATEWAY, SY1, and SY2 panels must be resident as they contain generic field definitions.

```
%INCLUDE(SYSTEM)
%INCLUDE(GATEWAY)
%INCLUDE(SYX)
```

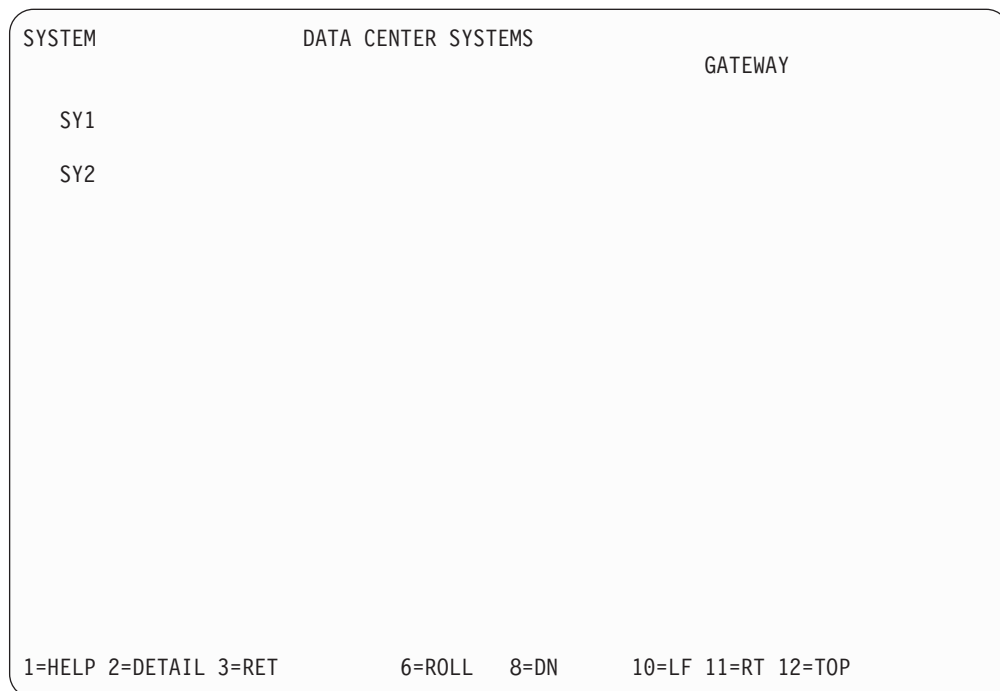
### Root Component Panel Definition

First, the root panel, named SYSTEM, is defined. Figure 19 shows the panel definition statements that define the SYSTEM panel. This panel is the default initial SDF panel as assigned in the SDF initialization parameter member, AOFINIT. Three panels are accessed by pressing the DOWN PF key (PF8), GATEWAY, SY1, and SY2. All status components are prefixed with the root component and are listed in the corresponding tree structure. Each STATUSFIELD (SF) statement is followed by the corresponding STATUSTEXT (ST) statement. Similarly, each TEXTFIELD (TF) statement is followed by the corresponding TEXTTEXT (TT) statement.

```
/* DEFINE SYSTEM STATUS PANEL
P(SYSTEM,24,80)
TF(01,02,10,WHITE,NORMAL)
TT(SYSTEM)
TF(01,25,57,WHITE,NORMAL)
TT(DATA CENTER SYSTEMS)
SF(SY1.SYSTEM,04,04,11,N,,SY1)
ST(SY1)
SF(SY2.SYSTEM,06,04,11,N,,SY2)
ST(SY2)
SF(SY1.GATEWAY,02,70,77,N,,GATEWAY)
ST GATEWAY
TF(24,01,48,T,NORMAL)
TT(1=HELP 2=DETAIL 3=RET 6=ROLL 8=DN)
TF(24,51,79,T,NORMAL)
TT(10=LF 11=RT 12=TOP)
EP
```

Figure 19. SDF Example: System Panel Definition Statements

This panel shows the layout defined by the statements in Figure 19:



### Status Component Panel Definition

Next, the panels for the status components, SY1 and SY2, are defined. These panels can be accessed by pressing the DOWN PF key (PF8) on the root component panel, after placing the cursor under the desired system name. They can also be accessed directly by entering SDF SY1 or SDF SY2 from the NetView NCCF command line, or entering SCREEN SY1 from within SDF.

Because these panels contain dynamic status elements, it is necessary for them to be made resident. This is done by referring to them in %INCLUDE statements in the main SDF panel definition member.

Figure 20 on page 322 shows a sample panel definition for panel SY1 and SY2.

## SDF Definitions

```
/* Panel definition statements for SY1/SY2 panel
P(&SDFROOT.,24,80,SYSTEM,SYSTEM)
TF(01,02,10,WHITE,NORMAL)
TT(&SDFROOT.)
TF(01,27,47,WHITE,NORMAL)
TT(&SDFROOT. SYSTEM STATUS)
SF(&SDFROOT..JES,04,16,24,N)
ST(JES)
SF(&SDFROOT..RMF,06,16,24,N)
ST(RMF)
SF(&SDFROOT..VTAM,08,16,24,N)
ST(VTAM)
SF(&SDFROOT..TSO,10,16,24,N)
ST(TSO)
SF(&SDFROOT..AOFAPPL,12.16,24,N)
ST(NetView)
SF(&SDFROOT..AOFSSI,14,18,28,N)
ST(NetView SSI)
SF(&SDFROOT..WTOR,4,45,50,N)
ST(WTORs:)
SF(&SDFROOT..WTOR,4,53,56,N,,c1)
SF(&SDFROOT..WTOR,4,59,67,N,,1)
SF(&SDFROOT..WTOR,5,53,56,N,,c2)
SF(&SDFROOT..WTOR,5,59,67,N,,2)
SF(&SDFROOT..WTOR,6,53,56,N,,c3)
SF(&SDFROOT..WTOR,6,59,67,N,,3)
SF(&SDFROOT..WTOR,7,53,56,N,,c4)
SF(&SDFROOT..WTOR,7,59,67,N,,4)
SF(&SDFROOT..APPLIC,9,45,57,N)
ST(Applications:)
SF(&SDFROOT..APPLIC,9,59,67,N,,1)
SF(&SDFROOT..APPLIC,10,59,67,N,,2)
SF(&SDFROOT..APPLIC,11,59,67,N,,3)
SF(&SDFROOT..APPLIC,12,59,67,N,,4)
SF(&SDFROOT..APPLIC,13,59,67,N,,5)
SF(&SDFROOT..APPLIC,14,59,67,N,,6)
PFK4('SDFCONF #ROOT,#COMPAPPL,#RV,#SID,#SNODE,#DATE,#TIME,#DA')
TF(24,01,79,T,NORMAL)
TT('1=HELP 2=DETAIL 3=RET 4=DELETE 6=ROLL 7=UP'
' 10=LF 11=RT 12=TOP')
EP
```

Figure 20. SDF Example: Status Component Panel Definition Statements for SY1 and SY2

SDF initialization uses the common global AOF\_AA0\_SDFROOT\_LISTn to generate as many panels as system names are found in the global for each panel that specifies P(&SDFROOT.,...).

Figure 21 on page 323 shows the layout defined by the statements in Figure 20 for system SY1.

**Note:** Three of the four available WTOR dynamic fields have been filled with the WTOR number and the name of the job that issued them. WTORs will appear whether or not their source is defined to SA z/OS.

If you do not define the common global AOF\_AA0\_SDFROOT\_LISTn it defaults to the local system name and SDF initialization members AOFPNLS and AOF TREE like the following REXX statement:

```
SAY "SDFROOT="cglobal(AOF_AA0_SDFROOT_LIST)
```

results in

```
SDFROOT=sysname
```

However you can specify other member names than the default member names, for example:

```
SYS1 SYS2/MYPNLS SYS3//MYTREE SYS4/MYPNLS/MYTREE
```

Assuming AOFPNLS and MYPNLS contain two INCLUDE statements.

```
AOFPNLS MYPNLS
%INCLUDE(ABC) DYNAMIC &INCLUDE(ABC) DYNAMIC
%INCLUDE(CDE) DYNAMIC &INCLUDE(XYZ) DYNAMIC
```

and all members define only one panel like "P(&SDFROOT.*member\_name*,...)" the following panels are generated when SDF initializes it:

For member ABC

```
SYS1ABC, SYS2ABC, SYS3ABC, SYS4ABC
```

For member CDE

```
SYS1CDE, , SYS3CDE
```

For member XYZ

```
 SYS2XYZ, SYS4XYZ
```

```
SY1 SY1 SYSTEM STATUS

 JES WTORS: 14 MSGPROC
 RMF 18 NETVIEW
 VTAM 22 MYJOB

 TSO Applications: MSGPROC
 NetView WTR00E
 NetView SSI IMS
 CICS
 ETC1
 ETC2

====>
1=HELP 2=DETAIL 3=RET 4=DELETE 6=ROLL 7=UP 10=LF 11=RT 12=TOP
```

Figure 21. Sample SY1 SDF Panel

The fields defined for JES, RMF, VTAM, TSO, NetView and the NetView SSI are static in that only the color of the predefined status text changes when the highest priority status descriptor that is queued for the underlying status component changes. The fields defining **WTORS:** and **Applications:** are also static, but do not refer to a specific subsystem. These fields will also assume the color of the highest priority status descriptor that is queued. The **WTORS:** field is green when no replies are outstanding due to the SDF tree definition for the underlying status component, SY1.WTOR. The remaining static fields will appear turquoise, or the EMPTYCOLOR that is defined in the AOFINIT NetView DSIPARM member.

The status fields following **WTORS:** and **Applications:** are dynamic in that both their content and color depend on the status descriptor that they represent. The

ability to select both the type of data and the status descriptor number that the data is obtained from allows generic status fields to be defined (see "STATUSFIELD" on page 311). This takes advantage of an SDF feature that allows the status descriptor to be queued under an alternate component should the primary status component not be defined in the SDF tree structure. For subsystems, the status component name is the subsystem name, and the alternate component is SUBSYS. WTORs are queued using the reply ID as the status component name, and WTOR as the alternate component name.

The use of generic field definitions has several advantages, and may considerably reduce the amount of maintenance required, particularly in large, multisystem environments. Using this method, the status components are displayed in priority order, so the most critical status subsystem is presented first. Also, if more subsystems are defined to SA z/OS than are defined on the panel, you will be notified of only the most critical situations. It is also possible to continue the list of statuses presented on additional panels if required.

You should note that using this method, subsystems do not always appear in the same position on the panel, which may make it difficult to find a specific subsystem. Also, some transient conditions can cause a subsystem to appear twice on the display. This can be eliminated by changing the SDF Status Detail definition to CLEAR=Y for the transient status definitions.

### SDF Initialization Parameters in AOFINIT

For this example, the default AOFINIT entries that are supplied with SA z/OS are used. For more information on setting SDF initialization parameters see Chapter 7, "SDF Initialization Parameters," on page 283.

```

SCREENSZ=10000,24,80
INITSCRN=SYSTEM
MAXOPS=10
PROPUP=YES
PROPDOWN=NO
TEMPERR=3
/* Status panel PF keys and description -----
PFK1=AOCHELP SDF
PFK2=DETAIL
PFK3=RETURN
PFK4=BACKWARD
PFK5=FORWARD
PFK6=ROLL
PFK7=UP
PFK8=DOWN
PFK9=
PFK10=LEFT
PFK11=RIGHT
PFK12=TOP
PFK13=AOCHELP SDF
PFK14=DETAIL
PFK15=RETURN
PFK16=BACKWARD
PFK17=FORWARD
PFK18=ROLL
PFK19=UP
PFK20=DOWN
PFK21=
PFK22=LEFT
PFK23=RIGHT
PFK24=TOP
/* Detail panel PF keys and description -----
DPFK1=AOCHELP SDF
DPFK2=

```



```

DPFK3=RETURN
/* Use the command below if you wish to delete without confirming
/* DPFK4=SDFCONF #ROOT,#COMPAPPL,#RV,#SID,#SNODE,#DATE,#TIME,#DA,VFY=NO
DPFK4=SDFCONF #ROOT,#COMPAPPL,#RV,#SID,#SNODE,#DATE,#TIME,#DA
DPFK5=
DPFK6=ROLL
DPFK7=UP
DPFK8=DOWN
DPFK9=
DPFK10=
DPFK11=BOT
DPFK12=TOP
DPFK13=AOCHELP SDF
DPFK14=
DPFK15=RETURN
/* Use the command below if you wish to delete without confirming
/* DPFK16=SDFCONF #ROOT,#COMPAPPL,#RV,#SID,#SNODE,#DATE,#TIME,#DA,VFY=NO
DPFK16=SDFCONF #ROOT,#COMPAPPL,#RV,#SID,#SNODE,#DATE,#TIME,#DA
DPFK17=
DPFK18=ROLL
DPFK19=UP
DPFK20=DOWN
DPFK21=
DPFK22=
DPFK23=BOT
DPFK24=TOP
DPFKDESC1=1=HELP 3=RETURN 4=DELETE 6=ROLL 7=UP 8=DOWN
DPFKDESC2=11=BOTTOM 12=TOP
/* Priority/color relationships (default values) -----
PRITBSZ=7
PRIORITY=1,199,RED
PRIORITY=200,299,PINK
PRIORITY=300,399,YELLOW
PRIORITY=400,499,TURQUOISE
PRIORITY=500,599,GREEN
PRIORITY=600,699,BLUE
DCOLOR=WHITE
EMPTYCOLOR=BLUE

```

**Note:** /\* denotes a comment field, where /\* must be followed by a blank.

## SDF Status Detail Definitions

Please refer to the Table in the section "Subsystem Colors and Priorities" of *IBM Tivoli System Automation for z/OS User's Guide* for a list of SDF default settings.

### Example Of A Large SDF Panel

This section shows an example of a large SDF panel that consolidates the tree definitions of SA z/OS default member INGTALL into a single display panel:

## Large SDF Panel

```
P(&SDFROOT.,62,160,SYSTEM,SYSTEM, ,*,*)
TF(01,02,05,PINK,REVERSE)
TT(&SDFROOT.)
TF(01,67,92,WHITE,NORMAL)
TT(A l l R e s o u r c e s)
/*- (1) -----
IF(03,02,BELOW)
/*- (2) -----
BODY(&SDFROOT..SUBSYS,06,35,*,1,02,27)
BH(05,,T,R,-)
BT(Subsystem Status)
CELL(02,12,N,C)
/*- (3) -----
BODY(&SDFROOT..GROUPS,38,47,*,1,02,27)
BH(37,L,S,T,R,+ -)
BT(Application Group Status)
CELL(02,12,N,C)
/*- (4) -----
BODY(&SDFROOT..MONITOR,50,*-3,*,1,02,27)
BH(49,C,N,T,R)
BT(Monitor Status)
CELL(02,12,N,C)
/*- (5) -----
BODY(&SDFROOT..WTOR,06,15,*,1,29,*)
BODYHEADER(05,LEFT,SCROLLINFO,T,R,-)
BODYTEXT(Jobname , Reply ID / Message text)
CELL(02,09,N) CELL(12,*,N,M)
```

*Figure 22. SDF definitions of a large display panel (Part 1 of 2). The sample includes all new definitions statements as well as the modified statements like INPUTFIELD.*

```

/*- (6) -----
BODY(&SDFROOT..CPMSGs,18,27,*,1,29,*)
BH(17,LE,SCROLL,TURQUOISE,REVERSE,+-)
BT(Jobname , Message text)
CELL(02,09,N) CELL(12,*,N,M)
/*- (7) -----
BODY(&SDFROOT..GATEWAY,30,39,*,1,29,*)
BH(29,LE,SCROLL,TURQUOISE,REVERSE,/-**-\)
BT(Gateway Message text)
CELL(02,07,N,C)
CELL(10,*,N,M)
/*- (8) -----
BODY(&SDFROOT..TAPE,42,46,*,1,29,*)
BH(41,LE,SCROLL,TURQUOISE,REVERSE,+)
BT(Device Message text)
CELL(02,05,N,C)
CELL(09,*,N,M)
/*- (9) -----
TF(49,30,33,T,N)
TT(Date)
TF(49,40,43,T,N)
TT(Time)
TF(49,50,58,T,N)
TT(Subsystem)
TF(49,62,73,T,N)
TT(Message text)
BODY(&SDFROOT..OPCERR,50,*-3,*,1,29,*)
BH(48,C,,TURQUOISE,REVERSE)
BT(Applications in ERROR)
CELL(02,09,N,D)
CELL(12,19,N,T)
CELL(22,32,N,C)
CELL(34,*,N,M)
/*-----
PFK3(UP)
PFK9(CLEAR)
PFK13('EXPLAIN #COMP,TARGET=#SNODE')
PFK14('SDFCONF #ROOT,#COMPAPPL,#RV,#SID,#SNODE,#DATE,#TIME,#DA')
PFK15('INGMSGs TARGET=&SDFROOT.')
PFK16('DISPGW TARGET=&SDFROOT.')
PFK17('SETSTATE #COMP,TARGET=#SNODE')
PFK18('INGVOTE #COMP/APL/#ROOT,TARGET=#SNODE')
PFK19('INGREQ #COMP/APL/#ROOT,TARGET=#SNODE')
PFK20('OPC TARGET=&SDFROOT.')
PFK22('DISPMTR #COMP,REQ=DETAIL,TARGET=#SNODE')
PFK23('INGLIST #COMP/APL/#ROOT,TARGET=#SNODE')
PFK24('INGINFO #COMP/APL/#ROOT,TARGET=#SNODE')
TF(*-1,02,13,T,NORMAL)
TT(1=Help)
TF(*-1,15,26,Y,NORMAL)
TT(2=Detail)
TF(*-1,28,159,T,NORMAL)
TT(3=Up 4=Backward ,
 5=Forward 6=Roll ,
 9=Clear 10=Previous 11=Next 12=Top)
TF(*,02,159,Y,NORMAL)
TT(13=EXPLAIN 14=SDFCONF 15=INGMSGs 16=DISPGW ,
 17=SETSTATE 18=INGVOTE 19=INGREQ 20=OPC ,
 22=DISPMTR 23=INGLIST 24=INGINFO)
EP

```

Figure 22. SDF definitions of a large display panel (Part 2 of 2). The sample includes all new definitions statements as well as the modified statements like INPUTFIELD.

# Large SDF Panel

```

x3270-4 EMEA
File Options
KEY4 All Reso
===>
No DOWN panel defined.
Subsystem Statu 1/60(113)
Jobname Reply ID / Message text -----
OMIIM2EZ IMS1DLS NETBTST4 031 DSI802A IPSNP REPLY WITH VALID NCCF
FFST SYSVSSI NETASYST 155 DSI802A IPSFP REPLY WITH VALID NCCF
DB2_DBM1 RMFGAT IMS941C4 158 DFS996I *IMS READY* M941
TCP/IP IMSCQS IMS942C4 206 DFS996I *IMS READY* M942
RACF IMS1DBRC
DLF SYSVAPPL
OMIIM2HD RMF
CPSMCMAS IMS1CTL
APPC SNA4MSTR
LLA JES2
OMIIM2HI DB2_IRLM
VLF SNA4DIST
OMIIM2RC OAM
OMIIM2CS DB2_MSTR
CICSTS SNA4IRLM
OMIIM2 DB2_DIST
CICSNC IMS2IRLM
OMIETE VTAM
ZFS SNA4DBM1
IMSRM CPSMWUI
OMIIOC0 RESOLVER
CICSDT TSD
IMSSCI ASCH
OMIICSUB CPSMLIST
OMVS AM2
IMSDM CICSKY45
OMIIC20 CICS1AR1
AM CICS1DOR
RRS CICS1FOR
IMSIRLM CICS1TOR

Application Grou 1/20(44)
SNA4_LITE_X CICS_SHARED
SNA_X BASE_USS
SNA3_X IMSA_CSLMGR
OMII_MVS OMII_CICS
BASE_SUPP OMII_GROUP
DB2 BASE_SYS
DB2_X IMS_SUPPORT
NETWORK SYSVIEW
SECURITY SNA4
LOOKASIDE SNA4_X

Gateway Message text /-**-\/-**-\/-**-\/-**-\/-**-\
IPSFMD ADF568I 10:55:26 : STATUS OF KEY4.IPSFMD OUT
IPSFND ADF568I 10:55:33 : STATUS OF KEY4.IPSFND OUT
IPSF00 ADF568I 10:55:33 : STATUS OF KEY4.IPSF00 OUT
IPSF70 ADF568I 10:55:41 : STATUS OF KEY4.IPSF70 OUT
IPUFA0 ADF568I 10:55:41 : STATUS OF KEY4.IPUFA0 OUT
IPUFB0 ADF568I 10:55:48 : STATUS OF KEY4.IPUFB0 OUT
IPUFCD ADF568I 10:55:55 : STATUS OF KEY4.IPUFCD OUT
IPUFD0 ADF568I 10:56:02 : STATUS OF KEY4.IPUFD0 OUT
IPUFG0 ADF568I 10:56:09 : STATUS OF KEY4.IPUFG0 OUT
IPUFH0 ADF568I 10:56:16 : STATUS OF KEY4.IPUFH0 OUT

Device Message text ++++++
Monitor Status Date Time Subsystem Message text
CICS1DB21 CI41TRABEND 11/18/11 11:02:38 XYZ TEST
C1TORDB22 CI42TRABEND
C1TORDB21 CICS1CON2
C1TORDB23 CICS44ABEND
IMS1DC CI44TRABEND
CICS45ABEND CICS44SOS
CI45TRABEND CICS1CON1
CI43TRABEND IMS2DC
CICS43SOS IMS1OLDS2
CICS41SOS IMS1OLDS1

1=Help 2=Detail 3=Up 4=Backward 5=Forward 6=Roll
13=EXPLAIN 14=SDFCONF 15=INGMSGS 16=DISPGW 17=SETSTATE 18=INGVOTE

```

Figure 23. Snapshot of the large display panel (left side) followed by Snapshot of the large panel (right side)

```

AVAMP
Sources
----- 11/18/11 11:08
----- 1/4(4)
SYSTEM OPERATOR COMMAND
SYSTEM OPERATOR COMMAND

----- 1/1(1)
S2CTL.DFS3257I HALTED - 3 ERRORS SINCE 15:13:42 ON 11/17/2011 - CRITICAL ERROR T

----- 1/10(35)
BOUND GATEWAY TO DOMAIN IPSFM IS INACTIVE - TARGET DOMAIN IPSFM IS NOT ACTIVE
BOUND GATEWAY TO DOMAIN IPSFN IS INACTIVE - TARGET DOMAIN IPSFN IS NOT ACTIVE
BOUND GATEWAY TO DOMAIN IPSFO IS INACTIVE - NO RESPONSE
BOUND GATEWAY TO DOMAIN IPSF7 IS INACTIVE - TARGET DOMAIN IPSF7 IS NOT ACTIVE
BOUND GATEWAY TO DOMAIN IPUFA IS INACTIVE - NO RESPONSE
BOUND GATEWAY TO DOMAIN IPUFB IS INACTIVE - TARGET DOMAIN IPUFB IS NOT ACTIVE
BOUND GATEWAY TO DOMAIN IPUFC IS INACTIVE - TARGET DOMAIN IPUFC IS NOT ACTIVE
BOUND GATEWAY TO DOMAIN IPUFD IS INACTIVE - TARGET DOMAIN IPUFD IS NOT ACTIVE
BOUND GATEWAY TO DOMAIN IPUFG IS INACTIVE - TARGET DOMAIN IPUFG IS NOT ACTIVE
BOUND GATEWAY TO DOMAIN IPUFH IS INACTIVE - TARGET DOMAIN IPUFH IS NOT ACTIVE

----- (0)

Applications in ERROR 1/1(1)

19=INGREQ 20=OPC 9=Clear 10=Previous 11=Next 12=Top
22=DISPMTR 23=INGLIST 24=ING INFO
FU0U3550 003/007

```

## Large SDF Panel

---

## Chapter 9. SDF Commands

---

### Using SDF Commands

SDF commands allow you to:

- to load one or more tree structures defined in a single member or to delete a single tree structure, see “SDFTREE” on page 336
- to load one or more panels defined in a single member or to delete a single panel, see “SDFPANEL” on page 335
- to initially display an SDF panel, “SDF” on page 333
- to move between panels, see “Navigation Commands” on page 337
- to move around the panels for each SDF display hierarchy, see “SCREEN” on page 333
- to delete an SDF record, see “SDFCONF” on page 334.

### Dynamically Loading Panels and Tree Structures

You can dynamically load panels and tree structures without restarting SDF. With this dynamic loading, you can load a small number of panels during initialization, and add or delete panel subsets when required during SDF operation. This can significantly reduce the number of panels kept resident at any one time.

When you are dynamically loading panels or tree structures, there must be a member in the NetView DSIPARM data set with the same name as the panel name or the root component in the tree structure. If not, a “not found” error message is generated.

**Note:** Only panels that are loaded with the SDFPANEL command are available to all logged-on SA z/OS operators. All others are loaded only for the operator that calls them.

### Dynamically Loading Panels

You can load panels dynamically in the following ways:

- With the SDFPANEL command, as described in “SDFPANEL” on page 335.
- With the SCREEN command, as described in “SCREEN” on page 333.
- When any of the following PANEL statement parameters call a panel not defined in AOFFPNLS, and a member with the same name as that panel is found in the NetView DSIPARM data set:

- *top\_panel\_name*
- *up\_panel\_name*
- *down\_panel\_name*
- *left\_panel\_name*
- *right\_panel\_name*

See “PANEL” on page 309 for the PANEL statement description.

**Performance hint:**

Dynamically loading panels reduces storage requirements. However, using the SCREEN command or PANEL statements that refer to the panels that are not defined in AOFPNLS can result in increased processor usage. For better performance, ensure the panels are included in the AOFPNLS member either directly or by an %INCLUDE.

## Dynamically Loading Tree Structures

You can load SDF tree structures dynamically with the SDFTREE command, as described in “SDFTREE” on page 336.

When you load a new tree structure to replace an existing one, any status descriptors with identical names in both tree structures are copied to the new tree structure.

## Dynamic Loading Example

Suppose you change the tree structure for root component SY1 and the panel named SY1SYS. The tree structure and panel definitions are maintained in separate members (instead of being directly coded in AOFTREE or AOFPNLS). Use the following commands to load the new definitions:

```
SDFTREE SY1,ADD
SDFPANEL SY1SYS,ADD
```

For more information, see “SDFTREE” on page 336 and “SDFPANEL” on page 335.

## Dynamic Loading Commands

Use the following commands to dynamically load SDF tree structures and panels, and to confirm that a panel was loaded:

**SDFTREE**

Load Tree Structure Definition Member

**SDFPANEL**

Load Panel Definition Member

**SCREEN**

Display a SDF Panel

When an error is detected while any of these dynamic loading commands is processing, no action is taken to change the existing tree structure or panel definitions. For example, if one of several panels defined or referenced by %INCLUDE statements in a panel definition member contains an error, none of the panels are placed into active use. Similarly, if an error is detected in a panel you attempt to load using the SCREEN command, the panel is not displayed.

## Verifying Dynamic Loading of Panels

Use the SCREEN command to verify that a panel was correctly loaded. See “SCREEN” on page 333 for the SCREEN command description.

You might want to create a test version of a panel you are modifying and display it using the SCREEN command to verify that your changes are correct. To do this:

1. Copy the existing panel definition member into another panel definition member.



2. Modify the panel definition statements in the new panel definition member. Use a different name for the panel on the PANEL statement.
3. Use the SCREEN command to verify that the changes to the panel are correct.
4. If you see anything in the displayed panel that should change, correct the panel definition statements.
5. Rename the panel to the name that is used for the production version of the panel. To do this, change the name specified on the PANEL statement.
6. Use the SDFPANEL command to load the new panel and put it into production. This SDFPANEL command causes the new panel to overwrite the old panel.

---

## SCREEN

### Purpose

The SCREEN command displays a specific SDF panel.

### Syntax

►►—SCREEN—*panel\_name*—◄◄

### Parameters

*panel\_name*

The name of the panel to be displayed. *panel\_name* is the name of the panel as it appears in the upper left hand corner of the screen.

### Restrictions and Limitations

SCREEN can be issued only within SDF.

### Usage

- If the specified panel is not in memory when the command is issued, the NetView DSIPARM data set is searched for a member name matching the specified panel name. If one is found, that member is loaded for the operator that the request was made from, and the panel defined in the member is displayed.
- If an error is detected in a panel you attempt to load using the SCREEN command, the panel is not displayed.
- If you plan to use the SCREEN command frequently in your SDF implementation, you might want to define a PF key that issues the SCREEN command.

### Examples

SCREEN SY1 displays the panel named SY1.

---

## SDF

### Purpose

Start the SDF dialog.





## SDFPANEL

|                   **cols**  
|                    The minimum number of columns. Values can range from 80 to 160. The  
|                    default is 80.

### Restrictions and Limitations

Panel definition members dynamically loaded by the SDFPANEL command are not reloaded when SDF is restarted. Only member AOFPNLS and any members referenced by %INCLUDE statements in AOFPNLS are reloaded. You must either add the panel definitions to AOFPNLS (using %INCLUDE statements) before SDF is restarted, or manually reload them using the SDFPANEL command after SDF is restarted.

### Usage

| If one of several panels defined or referenced by %INCLUDE statements in a panel  
| definition member contains an error, this panel is not placed in use.

| A panel which does not contain an error but defines the variable %SDFROOT  
| anywhere in its definition statements is replicated according to rules described in  
| "Status Component Panel Definition" on page 321 with the following exception. If  
| the member name that includes the panel does not occur in the  
| AOF\_AA0\_SDFROOT\_LISTn globals, the panel is replicated for each system name found  
| in the globals.

### Examples

SDFPANEL NEWPANEL,ADD loads member NEWPANEL into memory. This loading allows operators to access the panel defined in NEWPANEL.

| SDFPANEL VARPANEL,43,80 loads member VARPANEL into memory. SDF uses the  
| screen size 43x80 instead of the screen size defined at the initialization parameter  
| "SCREENSZ" on page 292. You may omit the column parameter. In this case the  
| minimum number of columns defaults to the number of columns defined in the  
| initialization parameter.

---

## SDFTREE

### Purpose

SDFTREE dynamically loads an SDF tree structure definition member from the NetView DSIPARM data set or deletes a tree member from system memory.

SDFTREE can be issued from a console.

### Syntax

To load or delete a tree structure definition member use the following syntax:

|                   → SDFTREE *tree\_member*, Add  
  DISKonly root\_component\_name,DElete →

### Parameters

*tree\_member*  
The name of the member containing the tree structure to load.

*root\_component\_name*

The name of the root component, which is the name that is used for level 1 in the tree structure that you want to delete. While you add a tree structure definition members by specifying a tree member name, you delete tree structure definition members by specifying a root component name.

**Add**

Specifies that you want to add the specified tree structure definition member.

**DISKonly**

Specifies that you want to add the specified tree structure member. However, any member previously loaded into storage is ignored

**DElete**

Specifies that you want to delete a tree structure definition member.

## Restrictions and Limitations

Tree structure definition members dynamically loaded by the SDFTREE command are not reloaded when SDF is restarted. When SDF is restarted, only members AOFREE and any members referenced by %INCLUDE statements in AOFREE are reloaded. You must either add the tree definitions to AOFREE (using %INCLUDE statements) before SDF is restarted, or manually reload them using the SDFTREE command after SDF is restarted.

## Usage

- When a new tree structure is loaded to replace an existing tree structure, the status descriptors of any status component with identical names in both trees are copied to the new tree.
- When an error is detected while this command is processing, no action is taken to change the existing tree structure.

## Examples

SDFTREE NEWTREE,ADD loads member NEWTREE into system memory. This loading allows operators to access the tree structure defined in NEWTREE.

---

## Navigation Commands

The following navigation commands are available within an SDF panel hierarchy:

**BACKWARD**

No action applies when entered in the detail panel.

When entered in the status panel, scrolls to the refreshed status panel in the stack.

**BOT**

When entered in the detail panel, displays the detail panel of the last status element in the chain.

No action applies when entered in the status panel.

**DETAIL**

No action applies when entered in the detail panel.

When entered in the status panel and the cursor is positioned at a status field or cell, displays the detail panel of the appropriate status element.

## Navigation Commands

### DOWN

When entered in a detail panel, displays the detail panel for the next status element in the chain.

When entered in a status panel:

- cursor in status element, displays the panel specified as "down" panel in the appropriate Status Field statement, if specified, otherwise the "down" panel specified in the PANEL statement,
- cursor not in status element, displays the panel specified as "down" panel in the PANEL statement,
- cursor in the body section,
  - ignores the command when the parameter IGNore has been specified in the PANEL statement,
  - displays the panel specified as "up|down" panel in the PANEL statement when the parameter IGNore has **not** been specified in the PANEL statement.

### FORWARD

No action applies when entered in the detail panel.

When entered in the status panel, scrolls to the refreshed panel that succeeds the panel in the stack.

### LEFT

No action when entered in the detail panel.

When entered in a status panel, displays the "left" panel specified in the PANEL statement.

### RETURN

When used in the detail panel, returns to the refreshed status panel.

When entered in the status panel, exits SDF.

### RIGHT

No action when entered in the detail panel.

When entered in a status panel, displays the "right" panel specified in the PANEL statement.

### SCREEN *panelname*

Displays the specified panel.

### TOP

When entered in a detail panel, displays the detail panel of the first status element in the chain.

When entered in the status panel, displays the "top" panel specified in the PANEL statement.

**UP** When entered in a detail panel, displays the detail panel for the previous status element in the chain.

When entered in a status panel:

- cursor in status element, displays the panel specified as "up" panel in the appropriate Status Field statement, if specified, otherwise the "up" panel specified in the PANEL statement,
- cursor not in status element, displays the panel specified as "up" panel in the PANEL statement,
- cursor in the body section,

- ignores the command when the parameter `IGNore` has been specified in the `PANEL` statement,
- displays the panel specified as "up | down" panel in the `PANEL` statement when the parameter `IGNore` has **not** been specified in the `PANEL` statement.

The navigation commands `DOWN`, `LEFT`, `RIGHT`, `SCREEN`, `TOP`, and `UP` put the panel name onto a stack. The navigation commands `BACKWARD` and `FORWARD` can be used to scroll thru the stack for displaying a panel. If the stack which holds at most 16 entries becomes full the first stack entry is removed before the current panel name is put onto the stack.

## Navigation Commands



---

## Appendix. 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

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan, Ltd.  
1623-14, Shimotsuruma, Yamato-shi  
Kanagawa 242-8502 Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:**

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement 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 Websites are provided for convenience only and do not in any manner serve as an endorsement of those Websites. The materials at those Websites are not part of the materials for this IBM product and use of those Websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who 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 Research & Development 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 licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

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

---

## Programming Interface Information

This book documents programming interfaces that allow the customer to write programs to obtain the services of System Automation for z/OS.

---

## Trademarks

IBM, the IBM logo, and [ibm.com](http://ibm.com)<sup>®</sup> are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.



---

## 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.** See automation configuration 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 automation control file, 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.

**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 NPDA 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) See Automation Manager Configuration. (2) The Auto Msg Classes entry type.

### **American Standard Code for Information Interchange (ASCII).**

A standard code used for information exchange among data processing systems, data communication systems, and associated equipment. ASCII uses a coded character set consisting of 7-bit coded characters (8-bit including parity check). The ASCII set consists of control characters and graphic characters. See also Extended Binary Coded Decimal Interchange Code.

**APF.** See authorized program facility.

**API.** See application programming interface.

**APPC.** See advanced program-to-program communication.

**application.** In SA z/OS, applications refer to z/OS subsystems, started tasks, or jobs that are automated and monitored by SA z/OS. On SNMP-capable processors, application can be used to refer to a subsystem or process.

**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.

**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.

**application programming interface (API).** An interface that allows an application program that is written in a high-level language to use specific data or functions of the operating system or another program.

**ApplicationGroup entry.** A construct, created with the customization dialogs, used to represent and contain policy for an application group.

**ARM.** See 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.** See American Standard Code for Information Interchange.

**ASF.** See automation status file.

**authorized program facility (APF).** A facility that permits identification of programs that are authorized to use restricted functions.

**automated console operations (ACO).** The use of an automated procedure to replace or simplify the action that an operator takes from a console in response to system or network events.

**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.

**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 SA z/OS customization dialogs must be used to build the automation configuration file. It consists of:

- The automation manager configuration file (AMC)
- The NetView automation table (AT)
- The NetView message revision table (MRT)
- 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 that SA z/OS is installed on. 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. See also "automation configuration file."

**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 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.

**automation status file (ASF).** 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

**Base Control Program (BCP).** A program that provides essential services for the MVS and z/OS operating systems. The program includes functions that



manage system resources. These functions include input/output, dispatch units of work, and the z/OS UNIX System Services kernel. See also Multiple Virtual Storage and z/OS.

**basic mode.** A central processor mode that does not use logical partitioning. Contrast with logically partitioned mode.

**BCP.** See Base Control Program.

**BCP Internal Interface.** Processor function of CMOS-390 and System z<sup>®</sup> processor families. It allows for 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.

| **blade.** A hardware unit that provides  
| application-specific services and components. The  
| consistent size and shape (or form factor) of each blade  
| allows it to fit in a BladeCenter chassis.

| **BladeCenter chassis.** A modular chassis that can  
| contain multiple blades, allowing the individual blades  
| to share resources such as management, switch, power,  
| and blower modules.

**BookManager<sup>®</sup>.** 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.

**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.

**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.

**CI.** See console integration.

**CICS/VS.** Customer Information Control System for Virtual Storage. See Customer Information Control System.

**CLIST.** See 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 processor control block.** An I/O operations internal control block that contains information about the command being processed.

**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 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.

**converted mode (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.

**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.** See central processor.

**CPC.** See central processor complex.

**CPC operations management commands.** A set of commands and responses for controlling the operation of System/390<sup>®</sup> CPCs.

**CPC subset.** All or part of a CPC. It contains the minimum *resource* to support a single control program.

**CPCB.** See command processor control block.

**CPU.** Central processing unit. Deprecated term for processor.

**cross-system coupling facility (XCF).** 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.** See converted mode.

## D

**DataPower X150z.** See IBM Websphere DataPower Integration Appliance X150 for zEnterprise (DataPower X150z).

**DASD.** See 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.** See double-byte character set.

**DCCF.** See disabled console communication facility.

**DCF.** See 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.

**device.** A piece of equipment. Devices can be workstations, printers, disk drives, tape units, remote systems or communications controllers. You can see information about all devices 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 that allows storage to be directly accessed, such as a disk drive.

**disabled console communication facility (DCCF).** A z/OS component that provides limited-function console communication during system recovery situations.

**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).

**display.** (1) To present information for viewing, usually on the screen of a workstation or on a hardcopy device. (2) Deprecated term for panel.

**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.

**Document Composition Facility (DCF).** An IBM licensed program used to format input to a printer.

**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 with 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.** See Extended Binary Coded Decimal Interchange Code.

**ECB.** See event control block.

**EMCS.** Extended multiple console support. See also multiple console support.

| **ensemble.** A collection of one or more zEnterprise  
| nodes (including any attached zBX) that are managed  
| as a single logical virtualized system by the Unified  
| Resource Manager, through the Hardware Management  
| Console.

| **ensemble member.** A zEnterprise node that has been  
| added to an ensemble.

**enterprise.** The composite of all operational entities, functions, and resources that form the total business concern and that require an information system.

**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.

**Enterprise Systems Architecture (ESA).** A hardware architecture that reduces the effort required for managing data sets and extends addressability for system, subsystem, and application functions.

**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.** See 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 System z 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, such that if all events of a trigger condition have occurred, a startup or shutdown of an application is performed.

**event control block (ECB).** A control block used to represent the status of an event.

**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 Binary Coded Decimal Interchange Code (EBCDIC).** A coded character set of 256 8-bit characters developed for the representation of textual data. See also American Standard Code for Information Interchange.

**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.

**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.

**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 (HMC).** A user  
| interface through which data center personnel  
| configure, control, monitor, and manage System z  
| hardware and software resources. The HMC  
| communicates with each central processor complex  
| (CPC) through the Support Element. On an IBM  
| zEnterprise 196 (z196), using the Unified Resource  
| Manager on the HMCs or Support Elements, personnel  
| can also create and manage an ensemble.

**Hardware Management Console Application (HWMCA).** A direct-manipulation object-oriented graphical user interface that provides a single point of control and single system image for hardware elements. The HWMCA provides 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 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 provides some level of abstraction from assembler language and independence from a particular type of machine. For the NetView program, the high-level languages are PL/I and C.

**HLL.** See high-level language.

**host (primary processor).** The processor that you enter a command at (also known as the *issuing processor*).

**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.

**HWMCA.** See Hardware Management Console Application.

| **Hypervisor.** A program that allows multiple instances  
| of operating systems or virtual servers to run  
| simultaneously on the same hardware device. A  
| hypervisor can run directly on the hardware, can run  
| within an operating system, or can be imbedded in  
| platform firmware. Examples of hypervisors include  
| PR/SM, z/VM, and PowerVM Enterprise Edition.

## I

| **IBM blade.** A customer-acquired, customer-installed  
| select blade to be managed by IBM zEnterprise Unified  
| Resource Manager. One example of an IBM blade is a  
| POWER7 blade.

| **IBM Smart Analyzer for DB2 for z/OS.** An optimizer  
| that processes certain types of data warehouse queries  
| for DB2 for z/OS.

| **IBM System z Application Assist Processor (zAAP).**  
| A specialized processor that provides a Java execution  
| environment, which enables Java-based web  
| applications to be integrated with core z/OS business  
| applications and backend database systems.

| **IBM System z Integrated Information Processor (zIIP).** A specialized processor that provides  
| computing capacity for selected data and transaction  
| processing workloads and for selected network  
| encryption workloads.

| **IBM Websphere DataPower Integration Appliance  
| X150 for zEnterprise (DataPower X150z).** A  
| purpose-built appliance that simplifies, helps secure,  
| and optimizes XML and Web services processing.



| **IBM Enterprise 196 (z196).** The newest generation of System z family of servers built on a new processor chip, with enhanced memory function and capacity, security, and on demand enhancements to support existing mainframe workloads and large scale consolidation.

| **IBM zEnterprise BladeCenter Extension (zBX).** A heterogeneous hardware infrastructure that consists of a BladeCenter chassis attached to an IBM zEnterprise 196 (z196). A BladeCenter chassis can contain IBM blades or optimizers.

| **IBM zEnterprise BladeCenter Extension (zBX) blade.** Generic name for all blade types supported in an IBM zEnterprise BladeCenter Extension (zBX). This term includes IBM blades and optimizers.

| **IBM zEnterprise System (zEnterprise).** A heterogeneous hardware infrastructure that can consist of an IBM zEnterprise 196 (z196) and an attached IBM zEnterprise BladeCenter Extension (zBX) Model 002, managed as a single logical virtualized system by the Unified Resource Manager.

| **IBM zEnterprise Unified Resource Manager.** Licensed Internal Code (LIC), also known as firmware, that is part of the Hardware Management Console. The Unified Resource Manager provides energy monitoring and management, goal-oriented policy management, increased security, virtual networking, and data management for the physical and logical resources of a given ensemble.

**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 Ops.** See I/O operations.

**I/O resource number.** Combination of channel path identifier (CHPID), device number, etc. See internal token.

**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.** See Information Management System.

**IMS/VS.** See 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 (IMS).** Any of several system environments available with a database manager and transaction processing that are capable of managing complex databases and terminal networks.

**Information Management System/Virtual Storage (IMS/VS).** A database/data communication (DB/DC) system that can manage complex databases and networks. Synonymous with Information Management System.

**INGEIO PROC.** The I/O operations default procedure name. It is part of the SYS1.PROCLIB.

**initial microprogram load.** The action of loading microprograms into computer storage.

**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 files and that prerequisite applications are functional.

**input/output configuration data set (IOCDs).** A configuration definition built by the I/O configuration program (IOCP) and stored on disk files associated with the processor controller.

**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. See also Time Sharing Option.

**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.** See input/output configuration data set.

**IOSP.** See input/output support processor.

**IPL.** See initial program load.

**ISPF.** See Interactive System Productivity Facility.

**ISPF console.** You log on to ISPF from this 3270-type console to use the runtime panels for I/O operations and SA z/OS customization panels.

**issuing host.** The base program that you enter a command for processing with. See primary host.

## J

**JCL.** See job control language.

**JES.** See job entry subsystem.

**JES2.** An MVS 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. See also job entry subsystem and JES3

**JES3.** An MVS 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 complexes that have several loosely coupled processing units, the JES3 program manages processors so that the global processor exercises centralized control over the local processors and distributes jobs to them using a common job queue. See also job entry subsystem and JES2.

**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).** An IBM licensed program that receives jobs into the system and processes all output data that is produced by jobs. In SA z/OS publications, JES refers to JES2 or JES3, unless otherwise stated. See also JES2 and JES3.

## K

**Kanji.** An ideographic character set used in Japanese. See also double-byte character set.

## L

**LAN.** See 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). See synchronous data link control. (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 on System z.** 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 that connects several devices in a limited area (such as a single building or campus) and that can be connected to a larger network.

**logical partition (LP).** A subset of the processor hardware that is defined to support an operating system. See also logically partitioned 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 and system services control point.

**logical unit 6.2 (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 peer relationship between session partners
- Efficient use of a session for multiple transactions
- A comprehensive end-to-end error processing

- A generic application program interface (API) consisting of structured verbs that are mapped to a product implementation

Synonym for advanced program-to-program communication.

**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.** See logical partition.

**LPAR.** See logically partitioned mode.

**LSN.** See logical switch number.

**LU.** See logical unit.

**LU 6.2.** See 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. See logical unit 6.2.

**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.

## M

**MAT.** Deprecated term for NetView automation table.

**MCA.** See Micro Channel architecture.

**MCS.** See multiple console support.

**member.** A specific function (one or more modules or 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.** See message processing facility.

**MPFLSTSA.** The MPFLST member that is built by SA z/OS.

**multi-MVS environment.** physical processing system that is capable of operating more than one MVS image. See also MVS image.

**multiple console support (MCS).** A feature of MVS that permits selective message routing to multiple consoles.

**Multiple Virtual Storage (MVS).** An IBM operating system that accesses multiple address spaces in virtual storage. The predecessor of z/OS.

**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 systems reside on one or more processors. Or one or more processors can communicate with programs on the other systems.

**MVS.** See Multiple Virtual Storage.

**MVS image.** A single occurrence of the MVS operating system that has the ability to process work. See also multi-MVS environment and single-MVS environment.

**MVS/ESA.** Multiple Virtual Storage/Enterprise Systems Architecture. See z/OS.

**MVS/JES2.** Multiple Virtual Storage/Job Entry System 2. A z/OS subsystem that receives jobs into the system, converts them to an 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.

## N

**NAU.** (1) See network addressable unit. (2) See network accessible unit.

**NCCF.** See Network Communications Control Facility.

**NCP.** (1) See network control program (general term). (2) See Network Control Program (an IBM licensed program). Its full name is Advanced Communications Function for the Network Control Program. Synonymous with ACF/NCP.

**NCP/token ring interconnection.** A function used by ACF/NCP to support token ring-attached SNA devices. NTRI also provides translation from token ring-attached SNA devices (PUs) to switched (dial-up) devices.

**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.

**NetView (NCCF) console.** A 3270-type console for NetView commands and runtime panels for system operations and processor operations.

**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 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 that NetView records events relating to NetView and SA z/OS activities in.

**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 message table.** See NetView automation table.

**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. See logical unit 6.2.

**NetView-NetView task (NNT).** The task that a cross-domain NetView operator session runs under. 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.

**network.** (1) An interconnected group of nodes. (2) In data processing, a user application network. See SNA network.

**network accessible unit (NAU).** In SNA networking, any device on the network that has a network address, including 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.

**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.

**network control program (NCP).** (1) A program that controls the operation of a communication controller. (2) A program used for requests and responses exchanged between physical units in a network for data flow control.

**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.

**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.

**NGMF.** Deprecated term for NetView Management Console.

**NGMF focal-point system.** Deprecated term for NMC focal point system.

**NIP.** See 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.** See 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.** See Network Problem Determination Application.

**NPDA focal-point system.** See focal point system.

**NTRI.** See 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.** See 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.** See Operations Planning and Control/Advanced.

**OPC/ESA.** See Operations Planning and Control/Enterprise Systems Architecture.

**Open Systems Adapter (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.

**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/Enterprise Systems Architecture (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.

| **optimizer.** A special-purpose hardware component or  
| appliance that can perform a limited set of specific  
| functions with optimized performance when compared  
| to a general-purpose processor. Because of its limited  
| set of functions, an optimizer is an integrated part of a  
| processing environment, rather than a stand-alone unit.  
| One example of an optimizer is the IBM Smart  
| Analytics Optimizer for DB2 for z/OS.

**OS.** See operating system.

**OSA.** See Open Systems Adapter.

**OST.** See 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, specifically 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, and 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.** A processor controller. Also known as the support processor or service processor in some processor families.

**PDB.** See policy database.

**PDS.** See 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 mode.

**POI.** See 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 automation definitions (automation policy) that the automation programmer specifies using the customization dialog is stored in the policy database. Also known as the PDB. See also automation policy.

**POR.** See power-on reset.

**port.** (1) System hardware that the I/O devices are attached to. (2) In 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 that data can enter or leave a data network through. (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.

**power-on reset (POR).** 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. See initial microprogram load.

**PP.** See physical partition.

**PPI.** See program to program interface.

**PPT.** See primary POI task.

**PR/SM™.** See Processor Resource/Systems Manager.

**primary host.** The base program that you enter a command for processing at.

**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.** (1) A device for processing data from programmed instructions. It may be part of another unit. (2) In a computer, the part that interprets and executes instructions. Two typical components of a processor are a control unit and an arithmetic logic unit.

**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 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 logically partitioned mode.

**ProcOps.** See 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. ProcOps 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 CICS, DB2, IMS, TWS (formerly called *features*).

**program operator interface (POI).** A NetView facility for receiving VTAM messages.

**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.

**PSM.** See ProcOps Service Machine.

**PU.** See physical unit.

## R

**RACF.** See Resource Access Control Facility.

**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 workstation from that user can log on to a domain from, that is, to the servers belonging to the domain, and use network resources. Users can access the shared resources and use the processing capability of the servers, thus reducing 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 Measurement Facility (RMF).** A feature of z/OS that measures selected areas of system activity and presents the data collected in the format of printed reports, System Management Facility (SMF) records, or display reports.

**Resource Object Data Manager (RODM).** In NetView for z/OS, a component that provides an in-memory 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.** Automation provided by SA z/OS 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 configuration file.

**Restructured Extended Executor (REXX).** A general-purpose, high-level, programming language, particularly suitable for EXEC procedures or programs for personal computing, used to write command lists.

**return code.** A code returned from a program used to influence the issuing of subsequent instructions.

**REXX.** See Restructured Extended Executor.

**REXX procedure.** A command list written with the Restructured Extended Executor (REXX), which is an interpretive language.

**RMF.** See Resource Measurement Facility.

**RODM.** See Resource Object Data Manager.

## S

**SAF.** See Security Authorization Facility.

**SA IOM.** See System Automation for Integrated Operations Management.

**SAPlex.** SAPlex or "SA z/OS Subplex" is a term used in conjunction with a sysplex. In fact, a SAPlex is a subset of a sysplex. However, it can also be a sysplex. For a detailed description, refer to "Using SA z/OS Subplexes" in *IBM Tivoli System Automation for z/OS Planning and Installation*.

**SA z/OS.** See 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 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.** See status display facility.

**SDLC.** See synchronous data link control.

**SDSF.** See 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.

**Security Authorization Facility (SAF).** An MVS interface with which programs can communicate with an external security manager, such as RACF.

**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 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 processor (SVP).** The name given to a processor controller on smaller System/370 processors.

**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).** A set of protocols for monitoring systems and devices in complex networks. Information about managed devices is defined and stored in a Management Information Base (MIB).

**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-MVS environment.** An environment that supports one MVS image. See also MVS image.

**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.** See service language command.

**SMP/E.** See System Modification Program/Extended.

**SNA.** See 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.** See Simple Network Management Protocol.

**solicited message.** An SA z/OS message that directly responds to a command. Contrast with unsolicited message.

**SSCP.** See system services control point.

**SSI.** See 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 that a subsystem or application is started.

**status.** The measure of the condition or availability of the resource.

**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.

**status focal-point system.** See focal point system.

**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.

**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.** See SAplex.

**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 (SSI).** 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.** See service processor.

**switch identifier.** The switch device number (swchdevn), the logical switch number (LSN) and the switch name

**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.

**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.** See 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<sup>®</sup> 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<sup>®</sup>.** 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.

**System/390 microprocessor cluster.** A configuration that consists of central processor complexes (CPCs) and may have one or more integrated coupling facilities.

**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.

## **T**

**TAF.** See 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 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.

**Tivoli Workload Scheduler (TWS).** A family of IBM licensed products that plan, execute and track jobs on several platforms and environments. The successor to OPC/A.

**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.** See 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.** See 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.** See Time-Sharing Option/Extended.

**TWS.** See Tivoli Workload Scheduler.

## U

**UCB.** See unit control block.

**unit control block (UCB).** A control block in common storage that describes the characteristics of a particular I/O device on the operating system 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.

**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.

## 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 Server.** A logical construct that appears to  
| comprise processor, memory, and I/O resources  
| conforming to a particular architecture. A virtual server  
| can support an operating system, associated  
| middleware, and applications. A hypervisor creates and  
| manages virtual servers.

| **Virtual Server Collection.** A set of virtual servers that  
| supports a workload. This set is not necessarily static.  
| The constituents of the collection at any given point are  
| determined by virtual servers involved in supporting  
| the workload at that time.

| **virtual Server Image.** A package containing metadata  
| that describes the system requirements, virtual storage  
| drives, and any goals and constraints for the virtual  
| machine (for example, isolation and availability). The  
| Open Virtual Machine Format (OVF) is a Distributed  
| Management Task Force (DMTF) standard that  
| describes a packaging format for virtual server images.

| **Virtual Server Image Capture.** The ability to store  
| metadata and disk images of an existing virtual server.  
| The metadata describes the virtual server storage,  
| network needs, goals and constraints. The captured  
| information is stored as a virtual server image that can  
| be referenced and used to create and deploy other  
| similar images.

| **Virtual Server Image Clone.** The ability to create an  
| identical copy (clone) of a virtual server image that can  
| be used to create a new similar virtual server.

**Virtual Storage Extended (VSE).** A system that consists of a basic operating system (VSE/Advanced Functions), and any IBM supplied and user-written programs required to meet the data processing needs of a user. VSE and the hardware that it controls form a complete computing system. Its current version is called VSE/ESA.

**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 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.

**VM/ESA®.** Virtual Machine/Enterprise Systems Architecture. Its current version is called z/VM.

**volume.** A direct access storage device (DASD) volume or a tape volume that serves a system in an SA z/OS enterprise.

**VSE.** See Virtual Storage Extended.

**VTAM.** See 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.

**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.** See write-to-operator.

**WTOR.** See 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.** See 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.** See extended recovery facility.

## Z

**z/OS.** An IBM mainframe operating system that uses 64-bit real storage. See also Base Control Program.

**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.

- | **z196.** See IBM Enterprise 196 (z196).
- | **zAAP.** See IBM System z Application Assist Processor (zAAP).
- | **zBX.** See IBM zEnterprise BladeCenter Extension (zBX).
- | **zBX blade.** See IBM zEnterprise BladeCenter Extension (zBX) blade.
- | **zCPC.** The physical collection of main storage, central processors, timers, and channels within a zEnterprise mainframe. Although this collection of hardware resources is part of the larger zEnterprise central processor complex, you can apply energy management policies to zCPC that are different from those that you apply to any attached IBM zEnterprise BladeCenter Extension (zBX) or blades. See also central processor complex.
- | **zIIP.** See IBM System z Integrated Information Processor (zIIP).
- | **zEnterprise.** See IBM zEnterprise System (zEnterprise).

## 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

&APPLPARMS  
ACFCMD command 10  
ACFREP command 25  
&COMPAPPL variable 289  
&DCOMP variable 289  
&QCOMP variable 289  
&RESAPPL variable 289

## Numerics

256-character allow or prohibit  
string 261, 263

## A

accessibility ix  
ACFCMD command 7  
ACFFQRY file manager command 16  
ACFREP command 21  
active message handler 29  
ACTIVMSG command 29  
address space management  
INGRCLUP command 126  
allow or prohibit attributes 263  
defining 261, 266  
AOCFLT command 31  
AOCGETCN command 33  
AOCMSG command 34  
AOCQRES command 39  
AOCQRY command 40  
AOCQRY task global variables 44  
AOCUPDT command 50  
AOFADMON monitoring routine 179  
AOFBJMON command  
See INGPJMON routine  
AOFAPMON monitoring routine 179  
AOFATMON monitoring routine 180  
AOFCPMSG command 55  
AOFCPSPM monitoring routine 180  
AOFEXCMD command 58  
AOFNCMON monitoring routine 181  
AOFPCILD.0 task global variable 64  
AOFPCILD.n task global variable 64  
AOFRACON command 60  
AOFRCMTR command 60  
AOFSET command 61  
AOFSHUTMOD global variable 168  
AOFSTREE 299  
AOFSTREE command 61  
AOFUXMON monitoring routine 181  
API  
assembler language CALLS 278  
description 193  
with REXX 273  
automation control file  
issuing commands from 7  
automation manager commands  
INGPOST 120  
INGRTCMD 138

Automation Table Function  
ING\$QRY 187  
AUTOTYPE task global variable 48

## B

BODY statement 302  
BODYHEADER statement 303  
BODYTEXT Statement 305

## C

captured message status definitions  
captured message colors 296  
captured message priorities 296  
cascaded switch  
FICON 195  
CDEMATCH command 66  
CELL statement 305  
CHKTHRES command 69  
CMDCNTHI task global variable 10  
code matching 66  
colors  
captured message 296  
gateway sessions 295  
groups 297  
in DISPSTAT 294  
in status display facility (SDF) 294  
monitor resource status 296  
Processor Operations 297  
ProcOps 297  
spool status 296  
TWS automation 297  
WTOR 296  
commands  
ACFCMD 7  
ACFFQRY 16  
ACFREP 21  
ACTIVMSG 29  
AOCFLT 31  
AOCGETCN 33  
AOCMSG 34  
AOCQRES 39  
AOCQRY 40  
AOCUPDT 50  
AOFADMON 179  
AOFAPMON 179  
AOFATMON 180  
AOFCPMSG 55  
AOFCPSPM 180  
AOFEXCMD 58  
AOFNCMON 181  
AOFRACON 60  
AOFRCMTR 60  
AOFSET 61  
AOFSTREE 61  
AOFUXMON 181  
CDEMATCH 66  
CHKTHRES 69  
DELETE FILE 209

commands (*continued*)

DISPSTAT  
colors 294  
FWDMSG 72  
HALTMSG 74  
INGALERT 76  
INGCLEAN 80  
INGCNTL 81  
INGCPSM 85  
INGDATA 87  
INGEXEC 91  
INGJLM 95  
INGLINK 98  
INGMON 101  
INGMTRAP 107  
INGOMX 108  
INGPJMON 182  
INGPOST 120  
INGPSMON 183  
INGQRY 123  
INGRCHCK 125  
INGRCLUP 126  
INGRDS 126  
INGRTCMD 138  
INGSIT 139  
INGSTOBS 141  
INGSTX 144  
INGTIMER 148  
INGUSS 150  
INGVARS 154  
INGVMON 185  
INGVSTOP 157  
INGVSTRT 158  
INGVTAM 159  
ISQMTSYS 185  
ISSUECMD 161  
MDFYSHUT 168  
OUTREP 169  
QUERY ENTITY CHP 210  
QUERY ENTITY CNTLUNIT 215  
QUERY ENTITY DEV 218  
QUERY ENTITY HOST 222  
QUERY ENTITY SWITCH 225  
QUERY FILE 228  
QUERY INTERFACE  
CNTLUNIT 229  
QUERY INTERFACE SWITCH 235  
QUERY RELATION CHP 243  
QUERY RELATION CNTLUNIT 244  
QUERY RELATION DEV 244  
QUERY RELATION HOST 246  
QUERY RELATION SWITCH 246  
QUERY SWITCH 247  
REMOVE CHP 250  
REMOVE DEV 254  
RESTORE CHP 250  
RESTORE DEV 254  
TERMMMSG 171  
WRITEFILE 259  
WRITEPORT 261  
WRITESWCH 266

communication mask 261, 263  
connectivity  
  defining 261  
connectivity, defining 263

## D

DATETIME statement 307  
DCOLOR parameter 283  
default status descriptor color 283  
define  
  color for SDF 286  
  I/O errors for SDF 293  
  maximum number of SDF  
  operators 288  
  SDF color/priority range 291  
  SDF color/priority relationship 290  
  SDF initial panel 287  
  SDF PF keys 284, 285, 288  
  SDF screen buffer size 292  
  status colors 287  
DELETE FILE command 209  
descriptor codes 35  
disability ix  
DPFKDESC1 parameter 285  
DPFKDESC2 parameter 285  
DPFKnn parameter 284

## E

EHKACTION task global variable 67  
EHKCMD task global variable 10  
EHKCMDTEXT task global variable 10  
EHKEXITNME task global variable 48  
EHKEXITRSN task global variable 48  
EHKRPY task global variable 21, 24  
EHKRPYHI task global variable 24  
EHKRPYTEXT task global variable 24  
EHKVARn task global variables 10, 25  
EMPTYCOLOR parameter 286  
ENDPANEL statement 308  
ERRCOLOR 287

## F

FALLBACK 295  
FICON cascaded switches 195  
FICON switches 195  
file manager commands  
  ACFFQRY 16  
filtering messages 31  
FWDMSG command 72

## G

gateway sessions  
  colors 295  
  priorities 295  
groups  
  colors 297  
  priorities 297

## H

HALTMSG command 74

## I

I/O operations  
  programming commands 193  
  safe switching 195  
I/O operations commands  
  DELETE FILE 209  
  QUERY ENTITY CHP 210  
  QUERY ENTITY CNTLUNIT 215  
  QUERY ENTITY DEV 218  
  QUERY ENTITY HOST 222  
  QUERY ENTITY SWITCH 225  
  QUERY FILE 228  
  QUERY INTERFACE  
  CNTLUNIT 229  
  QUERY INTERFACE SWITCH 235  
  QUERY RELATION CHP 243  
  QUERY RELATION CNTLUNIT 244  
  QUERY RELATION DEV 244  
  QUERY RELATION HOST 246  
  QUERY RELATION SWITCH 246  
  QUERY SWITCH 247  
  REMOVE CHP 250  
  REMOVE DEV 254  
  RESTORE CHP 250  
  RESTORE DEV 254  
  WRITEFILE 259  
  WRITEPORT 261  
  WRITESWCH 266  
ING\$QRY 187  
INGALERT command 76  
INGCLEAN command 80  
INGCNTL command 81  
INGCPSM command 85  
INGDATA command 87  
INGEXEC command 91  
INGJLM command 95  
INGLINK command 98  
INGMON command 101  
INGMTRAP command 107  
INGOMX command 108  
INGPJMOM monitoring routine 182  
INGPOST command 120  
INGPSMON monitoring routine 183  
INGQRY command 123  
INGRCHCK command 125  
INGRCLUP command 126  
INGRDS command 126  
INGRTCMD command 138  
INGSIT command 139  
INGSTOBS command 141  
INGSTX command 144  
INGTIMER command 148  
INGUSS command 150  
INGVARS command 154  
  line-mode output 156  
INGVMON monitoring routine 185  
INGVSTOP command 157  
INGVSTRT command 158  
INGVTAM command 159  
initialization parameters  
  DCOLOR 283  
  DPFKDESC1 285  
  DPFKDESC2 285  
  DPFKnn 284  
  EMPTYCOLOR 286  
  ERRCOLOR 287  
  INITSCRN 287

initialization parameters (*continued*)

  MAXOPS 288  
  PFKnn 288  
  PRIORITY 290  
  PRITBLSZ 291  
  PROPDOWN 292  
  PROPUP 292  
  SCREENSZ 292  
  TEMPERR 293  
INITSCRN parameter 287  
INPUTFIELD statement 308  
ISQMTSYS monitoring routine 185  
ISSUEACT command 161  
ISSUECMD command 161

## K

keyboard ix

## L

languages supported by the API  
  Assembler language 278  
  REXX 273  
  MVS REXX example 274  
load  
  QUERY FILE command 228  
load SDF tree structure 336  
LookAt message retrieval tool xvi

## M

mask 261, 263  
MAXOPS parameter 288  
MDFYSHUT command 168  
message forwarding and notification 72  
message generation and notification 34  
message retrieval tool, LookAt xvi  
modifying the current shutdown 168  
monitor resource status colors 296  
monitoring routine  
  INGSTOBS 141  
monitoring routines  
  AOFADMON 179  
  AOFAPMON 179  
  AOFATMON 180  
  AOFPCPSM 180  
  AOFNCMON 181  
  AOFUXMON 181  
  INGPJMOM 182  
  INGPSMON 183  
  INGVMON 185  
  ISQMTSYS 185  
MOVED 295  
MVS descriptor codes 35  
MVS REXX example 274

## N

NetView  
  Automation Table Function  
  ING\$QRY 187  
  DSIPARM member 299

## O

operating environment requirements 193  
OUTREP command 169  
outstanding WTORs  
  status display facility (SDF)  
    colors 296

## P

PANEL statement 309  
panels  
  Code Processing 68, 171  
  DISPACF 167  
  Message Processing 14  
parameter list  
  for I/O operations API 275  
PF key  
  defining for SDF 288  
PFKnn parameter 288  
PIB, see port information block 247  
port information block (PIB) 247  
priorities  
  captured message 296  
  gateway sessions 295  
  groups 297  
  monitor resource status 296  
  Processor Operations 297  
  ProcOps 297  
  spool status 296  
  TWS automation 297  
  WTOR 296  
priorities of subsystems in SDF 294  
priority  
  in status display facility (SDF) 294  
PRIORITY parameter 290  
PRITBSZ parameter 291  
programming commands, I/O  
  operations 193  
PROPDOWN parameter 292  
PROPUP parameter 292

## Q

QUERY ENTITY CHP command 210  
QUERY ENTITY CNTLUNIT  
  command 215  
QUERY ENTITY DEV command 218  
QUERY ENTITY HOST command 222  
QUERY ENTITY SWITCH  
  command 225  
QUERY FILE command 228  
QUERY INTERFACE CNTLUNIT  
  command 229  
QUERY INTERFACE SWITCH  
  command 235  
QUERY RELATION CHP command 243  
QUERY RELATION CNTLUNIT  
  command 244  
QUERY RELATION DEV command 244  
QUERY RELATION HOST  
  command 246  
QUERY RELATION SWITCH  
  command 246  
QUERY SWITCH command 247

## R

REMOVE CHP command 250  
REMOVE DEV command  
  RESTORE DEV command 254  
resource attribute, INGQRY  
  command 123  
RESTORE CHP command 250  
REXX coding instructions 273  
REXX EXEC  
  MVS example 274

## S

safe switching, I/O operations 195  
sample SDF, definition 318, 325  
save switch configuration  
  WRITEFILE command 259  
saved switch configuration  
  load file at IPL 228  
scheduling a command 58  
SCREEN command 333  
SCREENSZ parameter 292  
SDF  
  automation control file entry 283  
  initialization parameters 283  
  sample definition 318, 325  
  tree structure hierarchy 299  
SDF Command 333, 334  
SDF commands  
  SCREEN 333  
SDF Commands  
  SDF 333  
  SDFCONF 334  
SDF definition statements  
  AOFREE 299  
  BODY 302  
  CELL 305  
  DATETIME 307  
  DT 307  
  ENDPANEL 308  
  EP 308  
  IF 308  
  INPUTFIELD 308  
  P 309  
  PANEL 309  
  PFKnn 311  
  SDFPANEL 335  
  SDFTREE 336  
  SF 311  
  ST 314  
  STATUSFIELD 311  
  STATUSTEXT 314  
  TEXTFIELD 315  
  TEXTTEXT 317  
  TF 315  
  TT 317  
SDF Definition Statements  
  BH 303  
  BODYHEADER 303  
  BODYTEXT 305  
  BT 305  
SDF dialog  
  SCREEN 333  
SDF Dialog  
  SDF 333

SDF navigation commands

  BACKWARD 337  
  BOT 337  
  DETAIL 337  
  DOWN 338  
  FORWARD 338  
  LEFT 338  
  RETURN 338  
  RIGHT 338  
  SCREEN 338  
  TOP 338  
  UP 338  
shortcut keys ix  
spool status colors 296  
status  
  monitor resource, colors in SDF 296  
  spool, colors in SDF 296  
status definitions, captured message  
  captured message colors 296  
  captured message priorities 296  
status descriptor color 283  
status display facility (SDF)  
  colors 294  
  gateway colors 295  
  gateway priorities 295  
  monitor resource colors 296  
  monitor resource priorities 296  
  priorities 294  
  spool colors 296  
  spool priorities 296  
  WTOR colors 296  
  WTOR priorities 296  
status tree 292  
STATUSFIELD statement 311  
STATUSTEXT statement 314  
store  
  WRITEFILE command 259  
structure definitions  
  AOFREE 299  
  BH 303  
  BODY 302  
  BODYHEADER 303  
  BODYTEXT 305  
  BT 305  
  CELL 305  
  DATETIME 307  
  DT 307  
  ENDPANEL 308  
  EP 308  
  IF 308  
  INPUTFIELD 308  
  P 309  
  PANEL 309  
  PFKnn 311  
  SDFPANEL 335  
  SDFTREE 336  
  SF 311  
  ST 314  
  STATUSFIELD 311  
  STATUSTEXT 314  
  TEXTFIELD 315  
  TEXTTEXT 317  
  TF 315  
  TT 317  
SUBPAPPL task global variable 46  
SUBPASID task global variable 46



SUBPCATEGORY task global variable 46

SUBPCMDPFX task global variable 46

SUBPDESC task global variable 46

SUBPEXTSTART task global variable 46

SUBPEXTSTOP task global variable 46

SUBPFILE task global variable 46

SUBPFILTER task global variable 46

SUBPINFOLINK task global variable 46

SUBPIPLOPT task global variable 46

SUBPIPSTACK task global variable 46

SUBPJOB task global variable 46

SUBPJOBTYPE task global variable 46

SUBPMDATE task global variable 46

SUBPMTIME task global variable 47

SUBPOPER task global variable 47

SUBPPARENT task global variable 47

SUBPPATH task global variable 47

SUBPPID task global variable 47

SUBPPLEX task global variable 47

SUBPPORT task global variable 47

SUBPPROC task global variable 47

SUBPPROCESS 47

SUBPRSTOPT task global variable 47

SUBPSCHEDSS task global variable 47

SUBPSDATE task global variable 47

SUBPSESS task global variable 47

SUBPSHUTDLY task global variable 47

SUBPSPARM task global variable 47

SUBPSTARTYPE task global variable 47

SUBPSTAT task global variable 47

SUBPSTIME task global variable 47

SUBPSTOPTYPE task global variable 47

SUBPSTRTDLY task global variable 47

SUBPSUBCAT task global variable 47

SUBPSUBID task global variable 47

SUBPSUBTYPE task global variable 47

SUBPSYMBOLn task global variable 47

SUBPTERMDLY task global variable 47

SUBPTRANTY task global variable 47

SUBPTYPE task global variable 47

SUBPUSER task global variable 47

SUBPUSSJOB task global variable 47

SUBPWLMNAME task global variable 47

SUBPWTOR task global variable 47

SUBPxxxx task global variables 44

SUBSAPPL task global variable 45

SUBSASID task global variable 45

SUBSASSIST task global variable 48

SUBSCATEGORY task global variable 45

SUBSCMDPFX task global variable 45

SUBSDESC task global variable 45

SUBSEXTSTART task global variable 45

SUBSEXTSTOP task global variable 45

SUBSFILE task global variable 45

SUBSFILTER task global variable 45

SUBSINFOLINK task global variable 45

SUBSIPOPT task global variable 45

SUBSIPSTACK task global variable 45

SUBSJOB task global variable 45

SUBSJOBTYPE task global variable 45

SUBSMDATE task global variable 45

SUBSMTIME task global variable 45

SUBSOPER task global variable 45

SUBSPARENT task global variable 45

SUBSPATH task global variable 45

SUBSPID task global variable 45

SUBSPLEX task global variable 45

SUBSPORT task global variable 45

SUBSPROC task global variable 45

SUBSPROCESS task global variable 45

SUBSRSTOPT task global variable 45

SUBSSCHEDSS task global variable 45

SUBSSDATE task global variable 45

SUBSSESS task global variable 45

SUBSSHUTDLY task global variable 45

SUBSSPARM task global variable 45

SUBSSTARTYPE task global variable 45

SUBSSTAT task global variable 45

SUBSSTIME task global variable 45

SUBSSTOPTYPE task global variable 45

SUBSSTRTDLY task global variable 46

SUBSSUBCAT task global variable 46

SUBSSUBID task global variable 46

SUBSSUBTYPE task global variable 46

SUBSSYMBOLn task global variable 46

SUBSTERMDLY task global variable 46

SUBSTRANTY task global variable 46

SUBSTYPE task global variable 46

SUBSUSER task global variable 46

SUBSUSSJOB task global variable 46

SUBSWLMNAME task global variable 46, 47

SUBSWTOR task global variable 46

SUBSxxxx task global variables 44

subsystem colors and priorities in SDF 294

subsystem priorities in SDF 294

switch

- FICON 195
- FICON cascaded 195

switching, safe, I/O operations 195

syntax diagrams

- how to read 3

system hierarchy tree 61

system operations

- commands for programming 7

system operations commands

- INGCLEAN 80
- INGCNTL 81
- INGJLM 95
- INGPOST 120
- INGQRY 123
- INGRCHCK 125
- INGRCLUP 126
- INGRDS 126
- INGRTCMD 138
- INGUSS 150
- INGVSTOP 157
- INGVSTRT 158

## T

TEMPERR parameter 293

TERMMSG command 171

TEXTFIELD statement 315

TEXTTEXT statement 317

TWS automation

- colors 297
- priorities 297

## U

user-written programs

- calling the API 193

## W

WRITEFILE command 259

WRITEPORT command 261

- 256-character allow or prohibit string 263
- example 264

WRITESWCH command 266

WTORs

- status display facility (SDF) colors 296