System Automation for OS/390

Customizing and Programming

Version 2 Release 2

SC33-7035-04

System Automation for OS/390

Customizing and Programming

Version 2 Release 2

SC33-7035-04

Note!
Before using this information and the product it supports, be sure to read the general information under "Notices" on page xi.

Fourth Edition (April 2003)

This edition applies to System Automation for OS/390 (Program Number 5645-006) Version 2, Release 2, an IBM licensed program, and to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters.

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

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

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

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

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

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

Contents

Figures vii

Tables ix

Notices xi
Web Site Disclaimer xi
Programming Interface Information xi
Trademarks xii

About This Book xiii
Who Should Use This Book xiii
SA OS/390 User Groups xiii
Where to Find More Information xiv
The System Automation for OS/390 Library . . xiv
Related Product Information xv
Using LookAt to look up message explanations xvii
Accessing z/OS™ licensed documents on the Internet xvii

Chapter 1. A Guide to SA OS/390 Automation Table Samples 1
SA OS/390 Message Types 1
Automation Table Structure 1
Master Automation Tables 2
AOFMSGSY 3
SA OS/390 Message Presentation 4
Operator Cascades 8
Topology Manager 13
Integrating Automation Tables 13
Multiple Master Automation Tables 13
Using SA OS/390 %INCLUDE Fragments . . . 13

Chapter 2. How to Automate Your Resources 15
Using Automation Flags 15
Example 15
When SA OS/390 Checks Automation Flags . . . 16
The Automation Manager Global Automation Flag 16
Generic Automation Table Entries 17
Status Transition Messages 17
Abnormal End (Abend) Messages 18
Other Messages 19

Chapter 3. How to Add a New Application to Automation 21
Step 1: Defining an Application Policy Object . . . 21
Step 2: Defining Outstanding Reply Processing . . 21
Step 3: Building New System Operations Control Files 22
Step 4: Coding Entries for Application Messages in the MPF List 23
Step 5: Adding an Entry Calling ACTIVMSG to the NetView Message Automation Table 23
Adding an Entry Calling ACTIVMSG UP=YES to the NetView Message Automation Table 23
Step 6: Adding an Entry Calling TERMMSG to the NetView Message Automation Table 24
Adding an Entry Calling TERMMSG FINAL=YES to the NetView Message Automation Table . . . 24
Step 7: Adding SDF Entries for the Subsystem . . . 25
Updating SDF Tree Structure 25
Updating SDF Panels 26
Dynamically Loading SDF Tree Structure and Panels 26
Step 8: Enable the Application for Monitoring . . . 27
Step 9: Reloading Tables 27

Chapter 4. How to Automate USS Resources 29
SA OS/390 Enhancement for OS/390 UNIX System Services 29
Infrastructure Overview 29
Installing and Setting Up OS/390 UNIX Automation 30
Copying the SA OS/390 Command Server to the UNIX File System 30
Defining the UNIX Segments (OMVS) 31
BPX.JOBNAME 33
Customizing NetView 33
Restarting NetView 33
Customization of OS/390 UNIX Resources . . . 33
Debugging 44
Example: inetd 44
Hints and Tips 46
Trapping UNIX syslogd Messages 46

Chapter 5. How to Enable Enhanced Parallel Sysplex Automation 49
Enhanced Sysplex Functions 49
Managing Couple Data Sets 49
Managing the System Logger 50
Managing Coupling Facilities 52
Recovery Actions 55
The IBM Health Checker for z/OS and Sysplex 58
Hardware Validation 61
Miscellaneous 63
Enabling Parallel Sysplex Automation (Hardware Automation) 63
Step 1: Defining the Processor 63
Step 2: Using the Policy Item PROCESSOR INFO 64
Step 3: Defining Logical Partitions 64
Step 4: Defining the System 65
Step 5: Connecting the System to the Processor 65
Enabling Continuous Availability of Couple Data Sets 66
Defining the Policy Item SYSPLEX. 66
Enabling System Log Failure Recovery 67

Enabling WTO(R) Buffer Shortage Recovery . . . 69
Enabling System Removal 70
Step 1: Defining the Processor and System . . . 71
Step 2: Defining the Application with Application Type IMAGE 71
Step 3: Automating Messages IXC102A and IXC402D 72
Enabling Long Running Enqueues (ENQs) 74
Step 1: Defining Resources 75
Step 2: Making Job/ASID Definitions. 75
Step 3: Defining IEADMC*xx* Symbols. 76
Enabling Auxiliary Storage Shortage Recovery. . . 76
Step 1: Defining the Local Page Data Set. . . . 77
Step 2: Defining the Handling of Jobs 77
Customizing the IBM Health Checker for z/OS and Sysplex 78
Defining Common Automation Items. 80
Important Processor Operations Considerations . . 81

Chapter 6. How to Add a Message to Automation 83
Using Multiple NetView Message Automation Tables 83
Step 1: Automating the Response to the Message . . 83
Responding with a Command 84
Responding with a Reply. 84
Responding with Code Matching 84
Creating User Specific Message Processing . . . 85
Step 2: Adding an Entry in the NetView Message Automation Table 85
Messages Issued by an Application or MVS Component 85
Messages Issued by a Processor Operations Target System 86
Sample NetView Automation Table Entries . . . 87
Message ISQ211I. 88
Processor Operations Command Messages . . . 88
Testing Messages 89
Step 3: Assigning the Message to the Automation Environment 90
Adding a NetView Automation Operator ID . . 91
Step 4: Building the New Automation Definitions . 91
Step 5: Loading the Changed Automation Environment 91

Chapter 7. How to Write Your own Monitor Routines 93

Chapter 8. How to Create Automation Procedures 95
Programming Additional SA OS/390 Automation Procedures 95
How Generic Routines and Common Routines Are Used in Automation Procedures 95
How Automation Procedures Are Called 95
How Automation Procedures Are Structured . . . 96
Performing Initialization Processing 97
Determining Whether Automation Is Allowed . . 98
Performing Automation Processing 98
How to Make Your Automation Procedures Generic 102
Processor Operations Commands. 103
Developing Messages for Your Automation Procedures 103
Example AOCMSG Call 104
Adding NetView Message Automation Table Entries 104
Example Automation Procedure 104
Notes on the Automation Procedure Example 106
Installing Your Automation Procedures 107
Testing and Debugging Automation Procedures 107
The Assist Mode Facility 107
Using Assist Mode to Test Automation Procedures 108
Using AOCTRACE to Trace Automation Procedure Processing 110
NetView Testing and Debugging Facilities . . . 111
Where to Find More Testing Information . . . 112
Coding Your Own Information in the Automation Status File 112
Programming Recommendations 112
Global Variable Names 113

Chapter 9. How to Automate Processor Operations Controlled Resources. 115
Automating Processor Operations Resources of OS/390 Target Systems Using Proxy Definitions . . 115
The Concept. 115
Customizing Automation for Proxy Resources 116
Preparing Message Automation 118
Message Automation for IXC102A 118
Loss of Focal Point Authorization of a Support Element 118
How to Customize IXC102A Automation . . . 119
Automating Linux Console Messages 122
The Linux Console Connection to NetView . . 122
Linux Console Automation with Mixed Case Character Data 122
Security Considerations 123
Restrictions and Limitations 123

Chapter 10. SA OS/390 User Exits 125
Static Exits 126
AOFEXDEF 126
AOFEXI01 127
AOFEXI02 127
AOFEXI03 127
AOFEXI04 127
AOFEXINT 127
AOFEXSTA 128
AOFEXC00 129
AOFEXC01 129
AOFEXC02 130
AOFEXC03 131
AOFEXC04 132
Environmental Setup Exits 132
Flag Exits. 133
Parameters 134

Return Codes 135
Pseudo-Exits 136
Automation Control File Reload Permission Exit 136
Automation Control File Reload Action Exit . . 136
Subsystem Up at Initialization Commands . . 136
Testing Exits. 137
Customization Dialog Exits. 137
User Exits for BUILD Processing 137
User Exits for COPY Processing 138
User Exits for DELETE Processing 139
User Exits for CONVERT Processing 139
Invocation of Customization Dialog Exits . . . 140

Appendix A. Table of External Global Variables 141

Appendix B. Global Variables to Enable Advanced Automation 149

Appendix C. Customizing the Status Display Facility (SDF) 157
Overview of Status Display Facility 157
How SDF Works 157
Types of SDF Panels 157
Status Descriptors 158
SDF Tree Structures 159
How Status Descriptors Affect SDF 160
How SDF Helps Operations to Focus on Specific Problems 164
How SDF Panels Are Defined 164
Dynamically Loading Tree Structure and Panel Definition Members 165
Using SDF for Multiple Systems 165
SDF Components 166
How the SDF Task Is Started and Stopped . . 166
SDF Definition 167
Summary of SDF Definition Process 167
Step 1: Defining SDF Hierarchy 168
Step 2: Defining SDF Panels 169
Step 3: Customizing SDF Initialization Parameters 171
Step 4: Defining SDF in the Customization Dialog. 172

Glossary 173

Index 193

Figures

1. MAT Structure 2
2. Example of a WTORS Entry 21
3. Automation of OS/390 UNIX System Services 30
4. Job Example of Creating an OMVS Segment 32
5. Environment Setup Panel 34
6. Defining Application Type USS 35
7. Defining UNIX System Services Control Specifications 35
8. OS/390 UNIX Control Specification Panel for Type CLASS 36
9. OS/390 UNIX Control Specification Panel for Type INSTANCE 36
10. Application Automation Definition Panel 38
11. Startup Definition for a Process 42
12. Creating a Softlink 43
13. Stop Definitions for a Process 43
14. Delete a File 43
15. inetd Structure 44
16. Dependency Graphic 45
17. Example of a UNIX Message 47
18. Define New Entry Panel for Processors 64
19. Processor Information Panel 64
20. LPAR Definitions Panel 65
21. Define New Entry Panel for Sysplex Groups 65
22. Sysplex Policy Definition Panel for Sysplex Groups 67
23. Message Processing Panel 68
24. CMD Processing Panel 68
25. CMD Processing Panel 69
26. Message Processing Panel 70
27. Code Processing Panel 70
28. Definition of Application Type IMAGE 71
29. Application Automation Definition 72
30. Sample Panel for Defining IXC102A Automation 73
31. Sample Panel for Command Processing 74
32. Sample Panel for Code Processing 74
33. Long Running ENQ Resource Definition Panel for Sysplex Groups 75
34. Long Running ENQ Job/ASID Definitions Panel for Sysplex Groups 76
35. Long Running ENQ IEADMC*xx* Symbols Panel for Sysplex Groups 76
36. Local Page Data Set Recovery Panel for Sysplex Groups 77
37. Local Page Data Set Recovery Job Definition Panel for Sysplex Groups 78
38. UET Keyword-Data Specification Example 79
39. UET Keyword-Data Entry for L69 80
40. Sysplex Policy Definition Panel for Sysplex Groups 80
41. Automation Procedures for System Operations 96
42. Automation Procedures for Processor Operations 97
43. Skeleton of an Automation Procedure 103
44. SDF Detail Status Display Panel with Assist Mode 109
45. Message Processing Panel 119
46. CMD Processing Panel 120
47. Code Processing Panel 120
48. Minor Resource Selection Panel 121
49. Flag Automation Specification Panel (1) 122
50. SA OS/390 Exit Sequence during SA OS/390 Initialization 126
51. Example SDF Panels 158
52. Example SDF Tree Structure 160
53. Status Descriptors Chained to Status Components 162
54. Example Tree Structure Definition. 168
55. Example SDF Panel 170
56. Example Panel Definition Entry 170

Tables

1. System Automation for OS/390 Library xv
2. Related Products Books xv
3. Automation Flags: Typical Uses in SA OS/390 16
4. Externalized Common Global Variables 141
5. Global Variables to Enable Advanced Automation (CGLOBALS) 149
6. SDF Components 166
7. Panel Definition Entry Description 170

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

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

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

Web Site Disclaimer

Any pointers in this publication to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement. IBM accepts no responsibility for the content or use of non-IBM Web sites specifically mentioned in this publication or accessed through an IBM Web site that is mentioned in this publication.

Programming Interface Information

This publication primarily documents information that is NOT intended to be used as a Programming Interface of System Automation for OS/390.

This publication also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of System Automation for OS/390.

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

Programming Interface information

This section contains Programming Interface Information.

End of Programming Interface information

Trademarks

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

ACF/VTAM
APPN
CICS
DB2
DFSMS/MVS
ESCON
FICON
IBM
IMS
Library Reader
Micro Channel
MQSeries
Multiprise
MVS
MVS/ESA
Netfinity
Netfinity Manager
NetView
OpenEdition
OS/2
OS/390
Parallel Sysplex
RACF
RMF
Sysplex Timer
System/390
Tivoli
Tivoli Enterprise Console
VM/ESA
VSE/ESA
VTAM
z/OS
zSeries

The following terms are trademarks of other companies:

HeathKit
Heath Co., St. Joseph, MO

Spectracom
Spectracom Corp., Rochester, NY

Windows
Microsoft Corporation

Windows NT
Microsoft Corporation

Java Sun Microsystems, Inc.

UNIX UNIX is a registered trademark of The Open Group in the United States and other countries.

LINUX
LINUX is a registered trademark of Linus Torvalds and others.

About This Book

This book describes how to adapt your completed standard installation of System Automation for OS/390 (SA OS/390) as described in *System Automation for OS/390 Planning and Installation* to your environment. This book contains information on how to add new applications to automation or how to write your own automation procedures. Furthermore it contains information about how to add new messages of automated applications.

Who Should Use This Book

Installing, maintaining and using SA OS/390 is done by personnel from the user groups defined in "SA OS/390 User Groups". This book is primarily intended for automation programmers responsible for:

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

SA OS/390 User Groups

There are the following primary SA OS/390 user groups:

- **System programmers** are responsible for:

  product installation
  This is basically taking the product from tape and moving it into system libraries.

  system configuration
  This is basically allocating data sets and configuring individual systems to run the product. At the end of this process, a ready to use installation of the customization dialog and NetView is available to be used by further SA OS/390 user groups. Additional work may be required to help set up processor operations and the NMC workstation.

  maintenance
  This is the task of applying APARs to the base code and performing whatever system actions are required afterwards.

- **Automation programmers** are responsible for:

  policy migration
  If your enterprise uses policy for earlier SA OS/390 releases, this information needs to be migrated to the new release.

  policy definitions and distribution
  Using the customization dialog, automation programmers will, with help from specialist system programmers and operators, define the automation policy for the enterprise. This includes the definition of all automated resources and the individual definitions required for each resource. Thus they are creating the knowledge base that SA OS/390 will use to run their applications and systems. As the creation of this knowledge base can be done centrally on a system where the customization dialog is installed, the policy files that are created need to be distributed to the appropriate target systems before they can be used.

  automation customization
  This covers writing and modification of customer automation procedures

that run alongside SA OS/390. Existing automation procedures may need modifying to take account of changes that have been made with the current SA OS/390 release.

workstation setup and configuration
This task will also often fall upon the automation programmers. It is concerned with getting a workstation set up to run the SA OS/390 workstation, the NMC workstation and for the workstation user to be able to issue commands from it.

- **Operators** are the end users of the product, as it aims to simplify their jobs. They are responsible for:

  ensuring application availability
  This basically means starting applications when they need to be started, picking them up if they break down and stopping them when required.

  monitoring systems and applications
  Operators are responsible for detecting any problems that might occur with the systems or with the application software running upon them. When they find one, they will generally try a few standard fixes and then contact a system programmer.

  IPLs and shutdowns
  Operators are responsible for ensuring the orderly shutdown of systems and applications when they need to be stopped, and for their successful restarting afterwards.

  special operations
  This covers all of the things that the operators are asked to do that are not a part of their scheduled (or planned) activities. Examples might be keeping an application available until later in the day, changing the time that a particular event will happen, recycling applications to pick up fixes, or issuing special commands after an application has been started.

Where to Find More Information

The System Automation for OS/390 Library

The following table shows the information units in the System Automation for OS/390 library:

Table 1. System Automation for OS/390 Library

Title	Order Number
System Automation for OS/390 Planning and Installation	SC33-7038
System Automation for OS/390 Customizing and Programming	SC33-7035
System Automation for OS/390 Defining Automation Policy	SC33-7039
System Automation for OS/390 User's Guide	SC33-7040
System Automation for OS/390 Messages and Codes	SC33-7041
System Automation for OS/390 Operator's Commands	SC33-7042
System Automation for OS/390 Programmer's Reference	SC33-7043
System Automation for OS/390 CICS Automation Programmer's Reference and Operator's Guide	SC33-7044
System Automation for OS/390 IMS Automation Programmer's Reference and Operator's Guide	SC33-7045
System Automation for OS/390 OPC Automation Programmer's Reference and Operator's Guide	SC23-7046
System Automation for OS/390 Licensed Program Specifications	SC33-7037

The System Automation for OS/390 books (except Licensed Program Specifications) are also available on CD-ROM as part of the following collection kit:

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

SA OS/390 Homepage

For the latest news on SA OS/390, visit the SA OS/390 homepage at http://www.ibm.com/servers/eserver/zseries/software/sa

Related Product Information

The following table shows the books in the related product libraries that you may find useful for support of the SA OS/390 base program.

Table 2. Related Products Books

Title	Order Number
ISPF User's Guide	SC34-4484
ISPF Dialog Management Guide and Reference	SC34-4266
MVS/ESA MVS Configuration Program Guide and Reference	GC28-1817
MVS/ESA Planning: Dynamic I/O Configuration	GC28-1674
MVS/ESA Support for the Enterprise Systems Connection	GC28-1140
MVS/ESA Planning: APPC Management	GC28-1110
MVS/ESA Application Development Macro Reference	GC28-1822
OS/390: MVS System Commands	GC28-1781
MVS/ESA SPL Application Development Macro Reference	GC28-1857
OS/390 Hardware Configuration Definition: User's Guide	SC28-1848
OS/390 Information Roadmap	GC28-1727
OS/390 Information Transformation	GC28-1985
OS/390 Introduction and Release Guide	GC28-1725

Table 2. Related Products Books (continued)

Title	Order Number
OS/390 JES Commands Summary	GX22-0041
OS/390 Licensed Program Specifications	GC28-1728
OS/390 Printing Softcopy Books	S544-5354
OS/390 Starting Up a Sysplex	GC28-1779
OS/390 Up and Running!	GC28-1726
Planning for the 9032 Model 3 and 9033 Enterprise Systems Connection Director	SA26-6100
Resource Access Control Facility (RACF) Command Language Reference	SC28-0733
S/390 MVS Sysplex Overview – An Introduction to Data Sharing and Parallelism	GC23-1208
S/390 MVS Sysplex Systems Management	GC23-1209
S/390 Sysplex Hardware and Software Migration	GC23-1210
S/390 MVS Sysplex Application Migration	GC23-1211
S/390 Managing Your Processors	GC38-0452
Tivoli/Enterprise Console User's Guide Volume I	GC31-8334
Tivoli/Enterprise Console User's Guide Volume II	GC31-8335
Tivoli/Enterprise Console Event Integration Facility Guide	GC31-8337
Tivoli NetView for OS/390 Administration Reference	SC31-8222
Tivoli NetView for OS/390 Application Programming Guide	SC31-8223
Tivoli NetView for OS/390 APPN Topology and Accounting Agent	SC31-8224
Tivoli NetView for OS/390 Automation Guide	SC31-8225
Tivoli NetView for OS/390 AON Customization Guide	SC31-8662
Tivoli NetView for OS/390 AON User's Guide	GC31-8661
Tivoli NetView for OS/390 Bridge Implementation	SC31-8238
Tivoli NetView for OS/390 Command Reference Vol. 1	SC31-8227
Tivoli NetView for OS/390 Command Reference Vol. 2	SC31-8735
Tivoli NetView for OS/390 Customization Guide	SC31-8228
Tivoli NetView for OS/390 Customization: Using Assembler	SC31-8229
Tivoli NetView for OS/390 Customization: Using Pipes	SC31-8248
Tivoli NetView for OS/390 Customization: Using PL/I and C	SC31-8230
Tivoli NetView for OS/390 Customization: Using REXX and CLIST Language	SC31-8231
Tivoli NetView for OS/390 Data Mode Reference	SC31-8232
Tivoli NetView for OS/390 Installation: Getting Started	SC31-8767
Tivoli NetView for OS/390 Installation: Migration Guide	SC31-8768
Tivoli NetView for OS/390 Installation: Configuring Graphical Components	SC31-8770
Tivoli NetView for OS/390 Installation: Configuring Additional Components	SC31-8769
Tivoli NetView for OS/390 Messages and Codes	SC31-8237
Tivoli NetView for OS/390 MultiSystem Manager User's Guide	SC31-8607

Table 2. Related Products Books (continued)

Title	Order Number
Tivoli NetView for OS/390 NetView Management Console User's Guide	GC31-8665
Tivoli NetView for OS/390 User's Guide	SC31-8241
Tivoli NetView for OS/390 RODM and GMFHS Programming Guide	SC31-8233
Tivoli NetView for OS/390 Security Reference	SC31-8606
Tivoli NetView for OS/390 SNA Topology Manager and APPN Accounting Manager Implementation Guide	SC31-8239
Tivoli Management Platform Reference Guide	GC31-8324
TSO/E REXX/MVS User's Guide	SC28-1882
TSO/E REXX/MVS Reference	SC28-1883
VM/XA SP GCS Command and Macro Reference	SC23-0433
VSE/SP Unattended Node Support	SC33-6412
VTAM Messages and Codes	SC31-6493
VTAM Network Implementation Guide	SC31-6404
VTAM Network Implementation Guide	SC31-6434

Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most 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 access LookAt from the Internet at:
http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/ or from anywhere in z/OS or z/OS.e where you can access a TSO/E command line (for example, TSO/E prompt, ISPF, z/OS UNIX System Services running OMVS).

The LookAt Web site also features a mobile edition of LookAt for devices such as Pocket PCs, Palm OS, or Linux-based handhelds. So, if you have a handheld device with wireless access and an Internet browser, you can now access LookAt message information from almost anywhere.

To use LookAt as a TSO/E command, you must have LookAt installed on your host system. You can obtain the LookAt code for TSO/E from a disk on your *z/OS Collection* (SK3T-4269) or from the LookAt Web site's **Download** link.

Accessing z/OS™ licensed documents on the Internet

z/OS licensed documentation is available on the Internet in PDF format at the IBM Resource Link™ Web site at:

`http://www.ibm.com/servers/resourcelink`

Licensed documents are available only to customers with a z/OS license. Access to these documents requires an IBM Resource Link user ID and password, and a key code. With your z/OS order you received a Memo to Licensees, (GI10-0671), that includes this key code. [1]

1. z/OS.e™ customers received a Memo to Licensees, (GI10-0684) that includes this key code.

To obtain your IBM Resource Link user ID and password, log on to:
`http://www.ibm.com/servers/resourcelink`

To register for access to the z/OS licensed documents:

1. Sign in to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.

Note: You cannot access the z/OS licensed documents unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

Printed licensed documents are not available from IBM.

You can use the PDF format on either **z/OS Licensed Product Library CD-ROM** or IBM Resource Link to print licensed documents.

Chapter 1. A Guide to SA OS/390 Automation Table Samples

SA OS/390 provides an automation table which is described in the following. You will find it useful to have a copy of *Tivoli NetView for OS/390 Installation and Administration* because it explains the rules for the coding of the NetView automation table. Also, *Tivoli NetView for OS/390 Automation Guide* contains related information.

Note: *Tivoli NetView for OS/390 Installation and Administration* is non-entitled; that is, you do not automatically receive a copy when you order the NetView product. You must order this book separately.

SA OS/390 Message Types

The NetView message automation table determines how, in terms of highlighting attributes, various messages from SA OS/390 are displayed. To help you understand the table, the SA OS/390 message types are explained below.

Type I - Information
These messages are of an informational and non-urgent nature.

Type A - Immediate Action
These messages indicate that the operator has to do something in the near future to help SA OS/390 overcome a problem. SA OS/390 issues these if it needs operator intervention.

Type D - Immediate Decision
These messages are normally WTORs. They require a response from the operator (generally making a choice between a number of alternatives) before SA OS/390 will proceed with the actions necessary to implement that choice.

Type E - Eventual Action
These messages are issued when SA OS/390 detects an error in either its configuration data or command input. Generally operators must be aware of these errors and may be able to correct them, but sometimes you will need assistance from either your system programmers or SA OS/390 policy administrator.

Additionally, individual operators can select specific types of messages to be given additional highlighting when presented on their NCCF console. This is done by specifying that SA OS/390 hold the messages through its message notification processing. Held messages are displayed with a different set of actions from messages that are not held. These HOLD actions generally include HOLD(Y) to hold the message on the receiver's NCCF console.

Automation Table Structure

SA OS/390 provides a ready-to-use message automation table (MAT). To activate the MAT, perform the following steps:

- Define the MAT member INGMSG01 in the 'Automation Setup' policy of the system in the customization dialogs
- Build the ACF
- Restart NetView with the new configuration

A Guide to SA OS/390 Automation Table Samples

The SA OS/390 MAT contains:

- All/new entries for the SA OS/390 basic automation
- All entries for the IMS product automation
- All entries for the CICS product automation
- All entries for the OPC product automation
- All entries for the DB2 product automation
- All entries for the SAP High Availability Solution
- All entries for the XCF automation (msys for Operations)
- New entries required for the automation manager, MQ, RRS, TCP/IP, NFS, RACF, DFRMM
- User include fragments

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

Figure 1 shows the structure of the MAT:

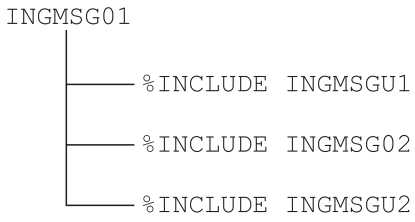


Figure 1. MAT Structure

For information about how to use the INCLUDE fragments that SA OS/390 provides, refer to "Using SA OS/390 %INCLUDE Fragments" on page 13.

The following fragments are used by the MAT:

Synonym Definitions
There is one fragment in this class, AOFMSGSY, which is used to initialize the various synonyms used throughout the rest of the table. SA OS/390 requires the synonyms to be suitably customized to reflect your environment.

SA OS/390 Functional Definitions
These definitions (located in INGMSG02) contain automation table statements for specific functions of SA OS/390. You should not change these statements. Any modifications can be made in INGMSGU1.

Application Automation Definitions
These fragments (all named AOFMxxxx) contain automation table statements related to the automation of either a single application (such as JES2) or a small group of closely related applications (such as VLF and LLA).

Master Automation Tables

This section discusses the three master automation tables that SA OS/390 provides.

INGMSG00
The message automation table INGMSG00 is used for SA OS/390 initialization. INGMSG00 is used during the initial ACF load. INGMSG00 does not have to be modified by the customer.

This table makes use of the synonyms defined in AOFMSGSY.

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

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

AOFMSGST
This is a table suitable for a NetView with a SA OS/390 Satellite installed.

AOFMSGSY

This automation table fragment contains a number of synonyms that must be appropriately set. It is used in most master automation tables to set up the environmental parameters for the other fragments. The AOFMSGSY member is supplied by SA OS/390 (in the SINGNPRM data set). You must customize it for each of your systems. The customized copy should be placed in the domain-specific data set for that system.

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

Synonym	Usage and Default
%AOFDOM%	This synonym should contain the domain ID of the SA OS/390 NetView on the system that it is automating. The synonym is used to screen messages to prevent the SA OS/390 on this machine from reacting to a message that originated on another machine. If not set correctly, your automation will fail. **Default** `&DOMAIN.` This is a default domain name used in a number of the samples.
%AOFSYS%	This synonym should contain the system name used in the last IPL of the system. It is used to screen messages to prevent the SA OS/390 on this machine from reacting to events that have occurred on other machines. It is important if you are running on a JES3 global or in a sysplex with EMCS consoles. If not set correctly, your automation will fail. **Default** `&SYSNAME.` This is a default system name used in a number of the samples.

SA OS/390 Message Presentation

The presentation of SA OS/390 messages (prefixed with AOF, ING, HSA, EVJ, EVE and EVI) under NetView is controlled by the message automation table. This uses a number of synonyms and task globals indicating your message display characteristics. The following synonyms determine the display characteristics for each type of message. There is one set for the normal presentation of the message (AOFNORMx) and a second set for the held presentation (AOFHOLDx).

%AOFHOLDI%

Usage

This synonym defines the actions taken for SA OS/390 information (type I) messages that are being held on your NCCF console. Given the number of informational messages that SA OS/390 produces you may find it beneficial HOLD(N) to stop them from being held even if the user has asked for them to be held.

Default

```
HOLD(Y) COLOR(GRE) XHILITE(REV)
```

This:

- Ensures that the message is held, and
- Causes the message to be displayed in reverse video green.

%AOFHOLDA%

Usage

This synonym defines the actions taken for SA OS/390 immediate action (type A) messages that are being held on your NCCF console. As a rule, you should specify HOLD(Y) in the action.

Default

```
HOLD(Y) COLOR(RED) XHILITE(REV) BEEP(Y)
```

This:

- Ensures that the message is held,
- Causes the message to be displayed in reverse video red, and
- Sounds the terminal alarm when the message is displayed.

%AOFHOLDD%

Usage

This synonym defines the actions taken for SA OS/390 decision (type D) messages that are being held on your NCCF console. As a rule, you should specify HOLD(Y) in the action.

Default

```
HOLD(N) COLOR(WHI) XHILITE(REV) BEEP(Y)
```

This:

- Ensures that the message is not held,

- Causes the message to be displayed in reverse video white, and
- Sounds the terminal alarm when the message is displayed.

%AOFHOLDE%

Usage

This synonym defines the actions taken for SA OS/390 eventual action (type E) messages that are being held on your NCCF console. As a rule, you should specify HOLD(Y) in the action.

Default

```
HOLD(N) COLOR(YEL) XHILITE(REV) BEEP(Y)
```

This:

- Ensures that the message is not held,
- Causes the message to be displayed in reverse video yellow, and
- Sounds the terminal alarm when the message is displayed.

%AOFHOLDW%

Usage

This synonym defines the actions taken for SA OS/390 wait state (type W) messages that are being held on your NCCF console. As a rule, you should specify HOLD(Y) in the action.

Default

```
HOLD(N) COLOR(PIN) XHILITE(REV) BEEP(Y)
```

This:

- Ensures that the message is not held,
- Causes the message to be displayed in reverse video pink, and
- Sounds the terminal alarm when the message is displayed.

This is the same as the actions for a non-held wait state message, but given the gravity of the situations where the message is issued this is not inappropriate.

%AOFJES3JOB%

Usage

The AOFMJES3 member traps an IEF404I message as the final termination message for the job that is running JES3. It uses this synonym to identify the correct IEF404I message. If you are running JES3 you should set this synonym to the name of your JES3 job.

Default

```
'JES3'
```

This is a plausible name for a job that runs JES3.

%AOFLOPAUTOx%

Usage

The synonyms %AOFLOPAUTO1% and %AOFLOPAUTO2% are defined for the autotasks AUTO1 and AUTO2 and are set to be a literal version of the autotasks' names. They are passed to CLIST AOFRDFOP to define the AOFEXCMD function routing globals for the AUTO1 and AUTO2 autotasks.

The AUTO1 and AUTO2 autotasks are part of the base NetView installation. If you have removed them, you should put them back in. If you have renamed them, you must change the value (not the synonym name) of the synonyms to reflect your changes.

Defaults

```
'AUTO1' and 'AUTO2'
```

These are the default autotask names. You need to change them only if you have renamed the tasks.

%AOFNORMI%

Usage

This synonym defines the actions taken for SA OS/390 information (type I) messages that are not being held on your NCCF console. As a rule, you should not specify HOLD(Y) in the action.

Default

```
HOLD(N) COLOR(GRE)
```

This:

- Ensures that the message is not held and
- Causes the message to be displayed in green.

%AOFNORMA%

Usage

This synonym defines the actions taken for SA OS/390 Immediate Action (type A) messages that are not being held on your NCCF console. As a rule, you should not specify HOLD(Y) in the action.

Default

```
HOLD(N) COLOR(RED) BEEP(Y)
```

This:

- Ensures that the message is not held,
- Causes the message to be displayed in red, and
- Sounds the terminal alarm when the message is displayed.

%AOFNORMD%

Usage

This synonym defines the actions taken for SA OS/390 Decision (type D) messages that are not being held on your NCCF console. You may find it beneficial to force these messages to be held.

Default

```
HOLD(N) COLOR(WHI) XHILITE(BLI)
```

This:

- Ensures that the message is not held and,
- Causes the message to be displayed in blinking white.

%AOFNORME%

Usage

This synonym defines the actions taken for SA OS/390 Eventual Action (type E) messages that are not being held on your NCCF console. As a rule, you should not specify HOLD(Y) in the action.

Default

```
HOLD(N) COLOR(YEL)
```

This:

- Ensures that the message is not held and,
- Causes the message to be displayed in yellow.

%AOFNORMW%

Usage

This synonym defines the actions taken for SA OS/390 Wait State (type W) messages that are not being held on your NCCF console. You may find it beneficial to force these messages to be held.

Default

```
HOLD(N) COLOR(PIN) XHILITE(REV) BEEP(Y)
```

This:

- Ensures that the message is not held,
- Causes the message to be displayed in reverse video pink, and
- Sounds the terminal alarm when the message is displayed.

%AOFVTAMJOB%

Usage

The IST020I message from VTAM, which SA OS/390 uses as a termination message for the VTAM application, does not have the job name for VTAM available either within the message or its extended attributes. This synonym is used to supply the job name on the TERMMSG call that is triggered from the IST020I message in AOFMVTAM.

Default

```
'VTAM'
```

This is a plausible name for a job that runs VTAM.

Operator Cascades

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

%AOFOPAUTO1%

Usage

This cascade is used to route commands to AUTO1. If you have renamed AUTO1 you must change the synonym.

Default

```
AUTO1
```

There is no backup for AUTO1. If it fails when it is needed, many other things will probably fail as well.

%AOFOPAUTO2%

Usage

This cascade is used to route commands to AUTO2. If you have renamed AUTO2 you must change this synonym.

Default

```
AUTO2 AUTO1
```

If AUTO2 is not active, AUTO1 does its work.

%AOFOPBASEOPER%

Usage

This cascade is used to send commands to BASEOPER. If you are not using the standard names for SA OS/390 autotasks you must change this synonym. BASEOPER is mainly defined as a fallback operator and has very little work directly routed to it.

Default

```
AUTBASE AUTO1
```

AUTBASE is the operator ID that SA OS/390 uses for BASEOPER in its other samples. If AUTBASE is not active, AUTO1 is tried.

%AOFOPSYSOPER%

Usage

This cascade is used to send commands to SYSOPER. If you are not using the standard names for SA OS/390 autotasks you must change this synonym. SYSOPER is mainly defined as a fallback operator and has very little work directly routed to it.

Default

```
AUTSYS AUTBASE AUTO1
```

AUTSYS is the operator ID that SA OS/390 uses for SYSOPER in its other samples.

%AOFOPMSGOPER%

Usage

This cascade is used to send commands to MSGOPER. If you are not using the standard names for SA OS/390 autotasks you must change this synonym. MSGOPER is mainly defined to respond to miscellaneous messages.

Default

```
AUTMSG AUTSYS AUTBASE AUTO1
```

AUTMSG is the operator ID that SA OS/390 uses for MSGOPER in its other samples.

%AOFOPNETOPER%

Usage

This cascade is used to send commands to NETOPER. If you are not using the standard names for SA OS/390 autotasks you must change this synonym. NETOPER is defined for VTAM automation.

Default

```
AUTNET1 AUTNET2 AUTSYS AUTBASE AUTO1
```

AUTNET1 and AUTNET2 are the operator IDs that SA OS/390 uses for NETOPER in its other samples. NETOPER is the only sample automation function to have a backup defined in the samples.

%AOFOPJESOPER%

Usage

This cascade is used to send commands to JESOPER. If you are not using the standard names for SA OS/390 autotasks you must change this synonym. JESOPER is mainly defined for JES automation.

Default

```
AUTJES AUTSYS AUTBASE AUTO1
```

AUTJES is the operator ID that SA OS/390 uses for JESOPER in its other samples.

%AOFOPMONOPER%

Usage

This cascade is used to send commands to MONOPER. If you are not using the standard names for SA OS/390 autotasks you must change this synonym. MONOPER is used for regular monitoring and subsystem startups.

Default

```
AUTMON AUTSYS AUTBASE AUTO1
```

AUTMON is the operator ID that SA OS/390 uses for MONOPER in its other samples.

%AOFOPRECOPER%

Usage

This cascade is used to send commands to RECOPER. If you are not using the standard names for SA OS/390 autotasks you must change this synonym. RECOPER is used for recovery processing.

Default

```
AUTREC AUTSYS AUTBASE AUTO1
```

AUTREC is the operator ID that SA OS/390 uses for RECOPER in its other samples.

%AOFOPSHUTOPER%

Usage

This cascade is used to send commands to SHUTOPER. If you are not using the standard names for SA OS/390 autotasks you must change this synonym. SHUTOPER coordinates automated shutdowns.

Default

```
AUTSHUT AUTSYS AUTBASE AUTO1
```

AUTSHUT is the operator ID that SA OS/390 uses for SHUTOPER in its other samples.

%AOFOPGSSOPER%

Usage

This cascade is used to send commands to GSSOPER. If you are not using the standard names for SA OS/390 autotasks you must change this synonym. GSSOPER is used for generic subsystem automation.

Default

```
* AUTGSS AUTSYS AUTBASE AUTO1
```

AUTGSS is the operator ID that SA OS/390 uses for GSSOPER in its other samples.

If you want to turn off the "ASSIGN BY JOBNAME" feature, that is, the advanced automation CGLOBAL variable *AOF_ASSIGN_JOBNAME* (see Appendix B, "Global Variables to Enable Advanced Automation", on page 149) has been set to 0, you must remove the asterisk (*), because this may cause serialization problems.

Note: NetView's ASSIGN-BY-JOBNAME command which occurs prior to the automation-table processing will only affect messages which are associated with an MVS job name.

%AOFOPWTORS%
Usage

This cascade is used to route commands concerning WTORS. If you are not using the standard names for SA OS/390 autotasks you must change this synonym. Its use ensures that all WTOR processing is done on the same task and this is serialized.

Note: This must be the same cascade as synonym %AOFOPSYSOPER%.

Default

```
AUTSYS AUTBASE AUTO1
```

This specifies that AUTSYS is to do all the WTOR processing.

%AOFOPGATOPER%
Usage

This cascade is used to route commands to this domain's gateway autotask. As the autotask name contains the domain ID you must modify this synonym.

Default

```
GATAOF01
```

AOF01 is the default domain used in the other samples. There is no backup as the gateway CLISTs expect to be running on GATOPER.

%AOFSIRTASK%
Usage

NetView has a CNMCSSIR task that handles communications between the main NetView task and its SSI address space. This synonym should be set to the name of the task. If the synonym is not set properly, SA OS/390 fails to initialize.

Default

```
&DOMAIN.SIR
```

%AOFARMPPI%
Usage

This synonym should contain the name of the NetView autotask that is running the PPI interface from SA OS/390 to OS/390. It is used to route commands from the NetView message automation table to the autotask.

Default

```
AOFARCAT
```

%AOFGMFHSWAIT%
Usage

The time interval SA OS/390 waits after GMFHS initialization is complete before issuing the command to update the RODM with the current application automation states. Following the issuing of message DUI4003I GMFHS NETWORK

CONFIGURATION INITIALIZED SUCCESSFULLY, GMFHS resets the color of all SA OS/390 icons to grey (unknown). To set the SA OS/390 icons' color to the current automation states after the initialization of GMFHS, SA OS/390 must wait and issue the update command AFTER GMFHS has reset the colors to grey.

Default

```
Default:  00:02:00
```

%AOFTECTASKQ%

Usage

This is the name of the autotask for sending SA OS/390 events to the Tivoli Enterprise Console (TEC) with quotes.

Default

```
''AUTOTEC''
```

%AOFTECTASK%

Usage

This is the name of the autotask for sending SA OS/390 events to the Tivoli Enterprise Console (TEC) without quotes. AOFTECTASK and AOFTECTASKQ must contain the same name (with and without quotes).

Default

```
AUTOTEC
```

%AOFTECPPI%

Usage

This is the NetView PPI Receiver ID of the GEM message adapter (with quotes).

Default

```
''IHSATEC''
```

%AOFTECMODE%

Usage

Event generation mode (with quotes). Possible values are:

LOCAL
The GEM message adapter is running on this system. LOCAL is valid for the local configuration and for the focal point in the distributed configuration.

REMOTE
The GEM message adapter is running on a remote automation focal point. SA OS/390 messages will be generated on this target system and forwarded to a remote automation focal point system. There is no local GEM message adapter which can process SA OS/390 messages. REMOTE is valid for the target system in a distributed configuration.

Default

```
''LOCAL''
```

Topology Manager

These synonyms are being used and defined in the AOFMSGST fragment.

%AOFOPTOPOMGR%

Usage

This is the name of the autotask that the SA OS/390 topology manager runs on this system.

Default

```
TPO&DOMAIN.
```

%AOFINITOPOCMD%

Usage

This is the command issued to initialize the SA OS/390 topology manager.

Default

```
INGTOPO INIT &DOMAIN.TPO
```

Integrating Automation Tables

If you have any user-written automation table statements which you still want to use, you must now combine your primary table with SA OS/390's. There are several approaches to achieve this.

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

Multiple Master Automation Tables

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

You need not modify the INGMSG01 automation table or any of the fragments, except AOFMSGSY. It is easy to maintain SA OS/390 automation table fragments. However, you have to watch for new messages. It is easy to maintain your entries, because they are independent from SA OS/390 entries.

Using SA OS/390 %INCLUDE Fragments

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

The fragment INGMSGU1 can be used for user entries. These entries have precedence over the SA OS/390 entries. The default INGMSGU1 is an empty member. The fragment INGMSG02 contains all entries provided by SA OS/390. It is being built automatically from an abstract source. It should not be modified.

The fragment INGMSGU2 can be used for all entries that SA OS/390 does not provide any entries for. The default INGMSGU2 is an empty member. During ACF

COLD/WARM start the MAT(s) is/are being loaded and write a listing to the DSILIST dataset. This enables the use of the NetView AUTOMAN command to monitor and manage the MAT(s). Make sure that the size of your DSILIST data set is sufficient to store these listings. Without these listings you can just monitor/manage the MATs using AUTOTBL.

An example output of AUTOTBL STATUS:

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

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

An example of the AUTOMAN panel:

```
EZLKATGB              AUTOMATION TABLE MANAGEMENT

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

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

It is easy to maintain SA OS/390 automation table fragments, However, you have to watch for new messages. It is easy to maintain your entries, because they are independent from SA OS/390 entries.

For information about the MAT structure, refer to “Automation Table Structure” on page 1.

Chapter 2. How to Automate Your Resources

This chapter contains information on how to customize your SA OS/390 installation by programming various routines and procedures. It describes various ways of how to adapt your installation to your requirements.

Using Automation Flags

SA OS/390 extended automation flags (automation flags for minor resources) give you the ability to control the automation for individual messages and status changes. You cannot use extended automation flags to stop a status change from occurring, but you can use them to stop commands or replies being issued in response to a change to a particular status. This facility can be disabled by setting the AOFMINORCHK advanced automation option to 0.

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

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

To define messages or status information as minor resources, use the customization dialog to edit the *Minor Resources* policy item of the appropriate *Application* policy object or the *MVS Component* policy object. See *System Automation for OS/390 Defining Automation Policy* for more information.

When an application is about to change to a new status, the status change routines (ACTIVMSG, HALTMSG and TERMMSG) check whether the new status has been defined as a minor resource for the application before they issue any commands associated with the status change. See "How Generic Routines and Common Routines Are Used in Automation Procedures" on page 95 and *System Automation for OS/390 Programmer's Reference* for more information about SA OS/390 routines.

The command and reply routines (ISSUECMD and ISSUEREP) check to see if the message ID of the message that triggered them is defined as a minor resource under the associated application (or under MVSESA for a system message).

Note: Calling either ISSUECMD or ISSUEREP with AUTOTYP=NOCHECK disables this checking, but as it causes a number of incongruities, this is not recommended.

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

Example

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

1. The TSO Start flag will be checked
2. If either the TSO Start flag is turned on or minor resource checking is enabled, the TSO.IKT001D Start flag is checked.

3. If the TSO.IKT001D Start flag is turned on, ISSUEREP issues any replies appropriate to the message.
4. If the TSO.IKT001D Start flag is turned off (even though the TSO Start flag may be turned on), SA OS/390 does not attempt to reply to the message.

When SA OS/390 Checks Automation Flags

This section describes how SA OS/390 uses automation flags. It provides background information to help you customize SA OS/390-provided automation and to help you write your own automation procedures.

Table 3 summarizes how SA OS/390 typically uses automation flags. SA OS/390 provides a common routine, AOCQRY, to perform automation flag checking. See *System Automation for OS/390 Programmer's Reference* for a description of AOCQRY.

Table 3. Automation Flags: Typical Uses in SA OS/390

Automation Flag	Typical Use In SA OS/390
Automation	Checked before any other automation flag to determine if overall automation for the resource is on or off. If it is off, none of the following flags will be checked.
Initstart	Checked after the SA OS/390 initialization for the first start of an application. If this is on, SA OS/390 will start the resource - provided its goal is to be available.
Recovery	Checked to determine whether to proceed with performing recovery actions other than restarting a resource.
Restart	If this is on, SA OS/390 checks whether or not the resource is eligible for restart.
Shutdown	Checked to determine whether to proceed with a shutdown for the specified resource.
Start	Checked after the initial application start command or commands are issued and additional commands or replies are issued for the subsystem, to determine if startup is to be automated. This flag can be used to control how much of the complete resource startup process is automated.

Note: SA OS/390 will invoke an exit only if it needs to in order to evaluate a flag. For example, if an exit is specified on a subsystem restart flag but the global SUBSYSTEM Automation flag is off, SA OS/390 does not invoke the exit when it checks the restart flag because the setting for the subsystem Automation flag (inherited from the SUBSYSTEM Automation flag) is off.

If the situation is reversed (exit for the subsystem Automation flag and the SUBSYSTEM Restart flag is off) the exit would also not be invoked. See "Flag Exits" on page 133 for more information on automation flag exits.

Do not rely on SA OS/390 to invoke an exit every time a flag is checked. You can only rely on SA OS/390 to invoke an exit before it concludes that a flag is turned on.

The Automation Manager Global Automation Flag

Using the INGLIST or the INGSET command (see *System Automation for OS/390 User's Guide* or *System Automation for OS/390 Operator's Commands*) you can set an

automation flag for the individual resources, which is checked by the automation manager before it sends any order to the automation agent to start or stop the specific resource.

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

Generic Automation Table Entries

The basic automation table contains a number of generic automation table entries that can reduce your automation table overhead considerably. These samples use some of the advanced features of SA OS/390 to make automating your applications as simple and reliable as possible.

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

```
IF MSGID = 'IEF40'. & DOMAINID = %AOFDOM% THEN BEGIN;
*
  IF (TOKEN(2) = 'BAT'. | TOKEN(2) = 'JCL'.)
    THEN DISPLAY(N) NETLOG(N);
*
  IF MSGID = 'IEF403I' & TOKEN(2) = SVJOB
    THEN EXEC(CMD('AOCFILT 'SVJOB' ACTIVMSG JOBNAME='SVJOB)
         ROUTE(ONE %AOFOPGSSOPER%));
*
  IF MSGID = 'IEF404I' & TOKEN(2) = SVJOB
    THEN EXEC(CMD('AOCFILT 'SVJOB' TERMMSG JOBNAME='SVJOB',FINAL=YES')
         ROUTE(ONE %AOFOPGSSOPER%));
*
  ALWAYS;
*
END;
```

Recommendation:
You should use the AOCFILT generic routine as a wrapper for all routines that are triggered from IEF4... messages for better performance. For more information about AOCFILT refer to *System Automation for OS/390 Programmer's Reference*.

Putting the generic entries inside a BEGIN...END section reduces the scanning cost for a non-matching message. The following messages are processed by the sample generic automation table entries.

Status Transition Messages

ACTIVE Messages

The IEF403I message is trapped. The job name is extracted from the second token and passed through to ACTIVMSG. It may be beneficial to block job names such as INIT or SYSLOG.

The benefit of using this entry is that you do not need to code an ACTIVE message for any of the applications.

FINAL TERMINATION Messages

The IEF404I message is trapped and the job name is extracted from the second token and passed through to TERMMSG with FINAL=YES specified. Again it may be beneficial to block common job names that are not applications.

The benefit of coding this is that you do not have to have a FINAL TERMINATION message coded for any application. With this entry and the ACTIVE message entry you need only code an UP message for minimal automation of an application, though it is recommended that you also code a TERMINATION message.

Abnormal End (Abend) Messages

IEF450I

The generic entry traps the IEF450I message, extracts the job name (from token 2), and the system and user abend codes. These are passed through as codes 1, 2, and 3 respectively to a TERMMSG call. TERMMSG performs a code match against 'system IEF450I' and expects one of 6 codes to be returned. By coding the 'system IEF450I' entries correctly you avoid having multiple traps for this message. Acceptable codes are STOPPING/STOPPED, ABENDING/ABENDED, and BREAKING/BROKEN.

STOPPING
The application is terminating normally.

ABENDING
The application is suffering a recoverable abend. SA OS/390 may attempt to restart it.

BREAKING
The application is suffering a non-recoverable abend. SA OS/390 does not attempt to restart it.

The past tense versions of these codes indicate to SA OS/390 that this is the last message it gets from that application (FINAL=YES). For more information on code matching, refer to "Responding with Code Matching" on page 84.

Example

If the following was coded under system message IEF450I codes:

Code 1	Code 2	Code 3	Action
*	S913	*	BREAKING
TASK23	S222	*	STOPPED
*	S222	*	ABENDED
*	S213	*	BREAKING
CICS®	*	U2301	BREAKING
*	*	*	ABENDING

The effects would be:

1. Any application that abnormally ended with an S913 error (a RACF violation) is placed into BREAKING (and then BROKEN when its final termination messages arrive) and is not restarted by SA OS/390 until a human operator intervenes.

2. If job TASK23 came down with an S222 error (an operator cancel), it would be treated as a normal termination message and a TERMMSG would process as if it had received the final termination message for the application. What you want in this case is an S222 not followed by an IEF404I.
3. If any other job came down with an S222 it would be placed into the ABENDING status and also processed as if its final termination message had been received.
4. If an application abends with an S213 error (data set not found) it would be put into the BREAKING status and would not be eligible for restart.
5. If a job starting with the letters CICS abnormally ended with a user abend code of U2301, it would also be put into BREAKING status.
6. Any other combination of job names and abend codes causes the application to be put into the ABENDING status.

IEF451I

This message indicates that a job is canceled. The job name and the condition code are extracted and passed to TERMMSG for a search against 'system IEF451I' as codes 1 and 2 respectively. The valid actions for the codes are as above.

IEF452I and IEFC452I

These messages indicate that there is a JCL error in the job procedure. TERMMSG is called with BREAK=YES, so SA OS/390 does not attempt to restart the application because this would most probably fail.

IEF453I

Another message indicating a JCL error. TERMMSG BREAK=YES is called as well.

IEF402I

This message indicates an abnormal end. TERMSG FINAL=YES is called.

Other Messages

SA OS/390 provides automation table entries for several products. Refer to the sample automation tables for more information.

Chapter 3. How to Add a New Application to Automation

This chapter describes the steps that are required to automate and monitor a new application by SA OS/390.

Step 1: Defining an Application Policy Object

To add a new application to SA OS/390, you must create and define a new *Application* policy object using the SA OS/390 customization dialog. With the customization dialog, you also define how the new application should be automated by SA OS/390. For example, you set automation flags for the application, or you specify startup or shutdown commands for this application, or you link the application into an application group. How to achieve this, is described in detail in *System Automation for OS/390 Defining Automation Policy*.

Step 2: Defining Outstanding Reply Processing

SA OS/390 remembers all outstanding Write-to-Operator Replies (WTORs) it receives if it does not reply to them immediately through ISSUEREP. Because some applications may have more than one WTOR at the same time, and not all WTORs are equally important, a method of classifying WTORs has been introduced.

When SA OS/390 receives a WTOR through OUTREP, ACTIVMSG, HALTMSG, TERMMSG, or ISSUEREP, and does not reply to it immediately, it performs a search of 'application WTORS', and then 'MVSESA WTORS', using the message ID as the first code and the job name as the second. This gives a two word code. The first word is the priority of the WTOR; the second word is the type of WTOR.

Performance:
Applications which frequently issue WTORs, should have an entry WTORS in their ***Message Processing*** panel which you reach by selecting the MESSAGES policy item (see Figure 2). This will improve the performance by reducing searches within the automation control file as mentioned above.

1)***Message Processing*** Panel:

```
CODE___   WTORS_____________________________                  3
          Classification of IMS WTORs_____________
```

2)***Code Processing*** Panel:

```
    Code 1          Code 2          Code 3          Value Returned
DFS996I________ *_______________ _______________ NORMAL PRI_____________________
DFS3139I_______ *_______________ _______________ NORMAL PRI_____________________
*_____________ *_______________ _______________ IMPORTANT SEC___________________
```

Figure 2. Example of a WTORS Entry

Priority	Meaning
NORMAL	This is an ordinary message and it does not indicate a problem. Displayed in SDF in GREEN (NWTOR status).
UNUSUAL	This might indicate a problem. It is not a WTOR that is normally outstanding. Displayed in SDF in YELLOW (UWTOR status). This is the default if a WTOR is not matched in either table.
IMPORTANT	This indicates a problem. It must be replied to promptly and may indicate more serious problems. Displayed in SDF in RED (IWTOR status). It may be abbreviated to IMPORT.
IGNORE	This tells SA OS/390 to ignore the WTOR (RWTOR status). The WTOR is not displayed on the SDF screen and is not recorded in the automation status file. This priority can only have a type of SEC.

Type	Meaning
PRI	This is the primary WTOR for the application. When SA OS/390 needs to issue a reply to the application but the reply number and/or message ID is not specified, such as on a shutdown pass, SA OS/390 responds to the last primary WTOR it received for the application. This is the default type. PRI is not applicable if the priority is IGNORE.
SEC	This is not the primary WTOR for the application. SA OS/390 does not reply to this WTOR if it has a primary, or even an older secondary WTOR recorded for the application. SEC is the default if the priority is IGNORE.

Note: The above priorities are also represented on the WTORs icon on the NMC workstation.

The mentioned colors are default SA OS/390 colors.

SA OS/390 uses a list to keep track of the WTORs for each application. New primary WTORs are added to the front of the list, and new secondary WTORs are added to the back of the list. When SA OS/390 needs to get a reply number for an application, it takes the first reply in the list. If a secondary WTOR has been received but a primary has not, SA OS/390 replies to the secondary WTOR. Generally it replies to the latest primary WTOR that is still outstanding or the earliest secondary WTOR that is still outstanding if there are no primary WTORs.

Step 3: Building New System Operations Control Files

When you finish defining the application in the customization dialog, build the new system operations control files (automation control file and automation manager configuration file, also called the automation configuration) from the updated policy database. See *System Automation for OS/390 Defining Automation Policy* for more information.

After you have completed this step and "Step 9: Reloading Tables" on page 27, the application is known to SA OS/390 and can therefore be automated according to the policy defined in "Step 1: Defining an Application Policy Object" on page 21.

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

Step 4: Coding Entries for Application Messages in the MPF List

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

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

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

Step 5: Adding an Entry Calling ACTIVMSG to the NetView Message Automation Table

When an application begins execution, use the ACTIVMSG generic routine to indicate that the application is either in status ACTIVE (initializing) or UP (operational). This is essential to the active monitoring process, particularly when several seconds elapse before the application becomes fully operational.

Generally, the IEF403I message issued by OS/390® may be used for this purpose, however an early message issued by the application itself may be a better choice. The ACTIVMSG command must be routed to the automation operator used to automate messages originated by the application, to avoid potential timing problems.

Note: If there is a generic entry in the NetView message automation table for IEF403I you will not need to add another entry for this application.

For more information on the ACTIVMSG and the AOCFILT generic routines (AOCFILT for performance improvements), see *System Automation for OS/390 Programmer's Reference*.

Adding an Entry Calling ACTIVMSG UP=YES to the NetView Message Automation Table

An application startup message, such as UP or READY, indicates that an application has completed initialization and is ready to run. When initialization completes, the application status must change and dependent applications must be started. Generic routine ACTIVMSG with parameter UP=YES performs these functions.

Observe several application startup operations or review console logs to find several startups. Determine the application startup message used in these startups. To use SA OS/390-provided startup automation, make sure the message you select is always displayed. For some products, startup messages vary from one startup to the next.

Code a NetView message automation table entry to call ACTIVMSG with parameter UP=YES when the startup message is received, for example:

```
IF MSGID = 'AHL031I' & DOMAINID = %AOFDOM%
THEN EXEC(CMD('ACTIVMSG UP=YES')  ROUTE(ONE %AOFOPGSSOPER%));
```

For more information on the ACTIVMSG and the AOCFILT generic routines (AOCFILT for performance improvements), see *System Automation for OS/390 Programmer's Reference*.

Step 6: Adding an Entry Calling TERMMSG to the NetView Message Automation Table

For recoverable application abend messages, you can use the TERMMSG generic routine. TERMMSG handles shutdown (termination) messages and status changes issued from applications during a shutdown process. TERMMSG sets the application status to ABENDING when a recoverable application abend message occurs. When TERMMSG is run with FINAL=YES, applications with ABENDING status are restarted, provided that restarts are allowed and the abend thresholds are not exceeded.

To activate the message handling process for the abend message, add a NetView message automation table entry to call TERMMSG when the abend message is received.

An example NetView message automation table entry for TERMMSG is:

```
IF MSGID = 'COF023I' & DOMAINID = %AOFDOM%
THEN EXEC(CMD('TERMMSG ABEND=YES')  ROUTE(ONE %AOFOPGSSOPER%));
```

For more information on the TERMMSG and the AOCFILT generic routines (AOCFILT for performance improvements), see *System Automation for OS/390 Programmer's Reference*.

Adding an Entry Calling TERMMSG FINAL=YES to the NetView Message Automation Table

An application shutdown message, such as DOWN or ENDED, indicates that the application has completed the shutdown process and can be restarted if necessary. When shutdown completes, the application status must change, an application restart must be performed if necessary, or the automation shutdown process must be informed that the application shutdown has completed. Generic routine TERMMSG with parameter FINAL=YES performs these functions.

Observe several application shutdown operations or review console logs to find several shutdowns. Determine the application shutdown message from these shutdowns. To use SA OS/390-provided shutdown automation, make sure the message you select is always displayed; some products have variations in the messages issued, depending on whether an abend or normal shutdown occurs.

Code a NetView message automation table entry to call TERMMSG with parameter FINAL=YES when the application shutdown message is received. The following is an example entry:

```
IF MSGID = 'IEF404I' & TOKEN(2)=SVJOB & DOMAINID = %AOFDOM%
THEN EXEC(CMD('TERMMSG FINAL=YES,JOBNAME='SVJOB) ROUTE(ONE %AOFOPGSSOPER%));
```

Note: If there is a generic entry in the NetView message automation table for IEF404I you will not need to add another entry for this application.

Some applications do not issue shutdown messages. In these cases, you may have to code an entry for a $HASP395 message (*jobname* ENDED). The following is an example entry for the $HASP395 message.

```
IF MSGID = '$HASP395' & TOKEN(2)=SVJOB & DOMAINID= %AOFDOM%
THEN EXEC(CMD('TERMMSG FINAL=YES,JOBNAME=' SVJOB) ROUTE(ONE %AOFOPGSSOPER%));
```

For more information on the TERMMSG generic routine, see *System Automation for OS/390 Programmer's Reference*.

Step 7: Adding SDF Entries for the Subsystem

If you want the application to appear in SDF status displays, you must update the SDF tree structure and SDF panels with information about the new application. This section shows how to add these tree structure and panel definitions to SDF, and how to dynamically add these definitions to SDF so that they are immediately reflected in SDF.

Note: Only OS/390 systems and resources can appear in SDF status.

Refer to *System Automation for OS/390 User's Guide* for complete details on defining SDF tree structure and panels.

Notes:

1. If generic SDF entries (xxx,APPLIC) are being used, you can skip this step. Generic entries are provided in the SA OS/390 samples.
2. This process for modifying SDF tree structure and panel definitions assumes the main tree structure definition member AOFTREE, and the main panel definition member, AOFPNLS, contain only %INCLUDE statements referencing other tree structure and panel definition members. Using only %INCLUDE statements in AOFTREE and AOFPNLS is recommended, and is the method used in SA OS/390-provided versions of AOFTREE and AOFPNLS.

Updating SDF Tree Structure

To update the SDF tree structure, modify the appropriate NetView DSIPARM data set member containing SDF tree structure definitions to add an entry for the new application. This data set member must be referenced by a %INCLUDE statement in the main tree structure definition member, AOFTREE. In the tree structure definition, specify the level number and status component name to be used for the new application.

For example, suppose you want to add application GTF to the tree structure definition data set member SY1. Before you modify it, data set member SY1 looks like this:

```
1 SY1
  2 SUBSYS
     3 AOFAPPL
       4 AOFSSI
     3 JES
     3 VTAM
     3 TSO
     3 RMF
  2 GATEWAY
```

Subsystem GTF should be at the same level in the tree structure as the other subsystems, which is 3. You decide to assign a status component name of GTF to the GTF subsystem.

The modified SY1 data set member looks like this:

```
1 SY1
  2 SUBSYS
     3 AOFAPPL
       4 AOFSSI
     3 JES
     3 VTAM
     3 TSO
     3 RMF
     3 GTF
  2 GATEWAY
```

See *System Automation for OS/390 User's Guide* for more details on defining SDF tree structures, and *System Automation for OS/390 Programmer's Reference* for a description of AOFTREE entries.

Updating SDF Panels

To update SDF panels, decide which SDF status panel will display the application entry and modify the panel definition statements for that panel in the appropriate NetView DSIPARM data set member. This data set member must be referenced by a %INCLUDE statement in the main panel definition member, AOFPNLS.

For example, suppose you want to add the GTF subsystem to panel SY1PNLS. To do this, you add the following STATUSFIELD and STATUSTEXT entries for the GTF subsystem to the NetView DSIPARM data set member SY1PNLS:

```
STATUSFIELD(SY1.GTF,06,31,33,N,,)
STATUSTEXT(GTF)
```

For more information on how to define panels in AOFPNLS, refer to *System Automation for OS/390 User's Guide*.

Dynamically Loading SDF Tree Structure and Panels

You can load the changed SDF tree structure and panel definitions dynamically. This allows you to activate the changed SDF panels without restarting SDF.

To load the changed tree structure definition dynamically, use the SDFTREE command.

For example, to load the modified SY1 data set member containing the tree structure definition for the new GTF subsystem, type the following from a NetView console:

```
SDFTREE SY1,ADD
```

See *System Automation for OS/390 Programmer's Reference* for more information on the SDFTREE command.

To load the changed panel definition dynamically, use the SDFPANEL command. For example, to load the panel definition member SY1PNLS containing panel definition changes for the new GTF subsystem, type the following from a NetView console:

```
SDFPANEL SY1PNLS,ADD
```

See *System Automation for OS/390 Programmer's Reference* for more information on the SDFPANEL command.

Note: This process for dynamically loading SDF tree structure and panel definitions assumes the main tree structure definition member, AOFTREE,

and the main panel definition member, AOFPNLS, contain only %INCLUDE statements referencing other tree structure and panel definition members. Using only %INCLUDE statements in AOFTREE and AOFPNLS is recommended, and is the method used in SA OS/390-provided versions of AOFTREE and AOFPNLS.

When SDF is restarted, only members AOFTREE and AOFPNLS are loaded. Members loaded with SDFTREE or SDFPANEL commands are not reloaded during AOFTDDF initialization. You must either add the new panel and tree definitions to AOFTREE and AOFPNLS before SDF is started (using %INCLUDE statements in AOFTREE and AOFPNLS) or manually reload them using the SDFTREE or SDFPANEL commands after SDF is started.

Step 8: Enable the Application for Monitoring

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

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

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

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

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

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

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

Note: By default, the application will be included on NMC within the automatically generated views representing the application groups that it is a member of.

Step 9: Reloading Tables

Reload the MPF list, NetView message automation table, and automation control file to enable automation of the application.

To reload the MPF list, type the following command:

- from the OS/390 console:

  `SET MPF=xx`
- from a NetView console using the MVS prefix:

  `MVS SET MPF=xx`

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

To reload a NetView message automation table, either recycle NetView or type the following from a NetView console:

```
AUTOTBL MEMBER=msgtable, SWAP,NAME=msgtable
```

where *msgtable* is the name of the NetView message automation table to load.

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

```
INGAMS REFRESH
```

command and specify a data set name or an * which means: reuse the current one.

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

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

Chapter 4. How to Automate USS Resources

SA OS/390 Enhancement for OS/390 UNIX System Services

This enhancement better integrates applications, which run partly or totally in OS/390 UNIX System Services (OS/390 UNIX), into System Automation for OS/390 2.2

It is an OS/390 UNIX System Services automation which uses the functionality of SA OS/390 2.2 to automate OS/390 UNIX applications.

The following functions are now supported by SA OS/390 2.2 for OS/390 UNIX applications:

- Starting and stopping of applications
- Monitoring of:
  - Processes (represented by the command or path and user ID)
  - TCP Ports
  - Files and file systems
  - Generic User Monitoring (the user supplies a OS/390 UNIX monitoring routine or script)
- Using an API for execution of OS/390 UNIX commands. (INGUSS command)

Infrastructure Overview

The OS/390 UNIX resources that should be automated must run in the OS/390 UNIX of an OS/390 system that is already automated by SA OS/390. From the automation manager's perspective the NetView agent of this system is responsible for the OS/390 UNIX resources.

For command execution through INGUSS or user-defined monitoring, an OS/390 UNIX program (provided by SA OS/390) is directly invoked by SA OS/390. This program (ingccmd) executes UNIX commands and runs when started by SA OS/390 with the jobname INGCUNIX. ingccmd is the extension of the NetView based agent into OS/390 UNIX. To monitor the standard OS/390 UNIX resources (process, port, or file) an SA OS/390 internal routine is started. Figure 3 on page 30 illustrates this structure:

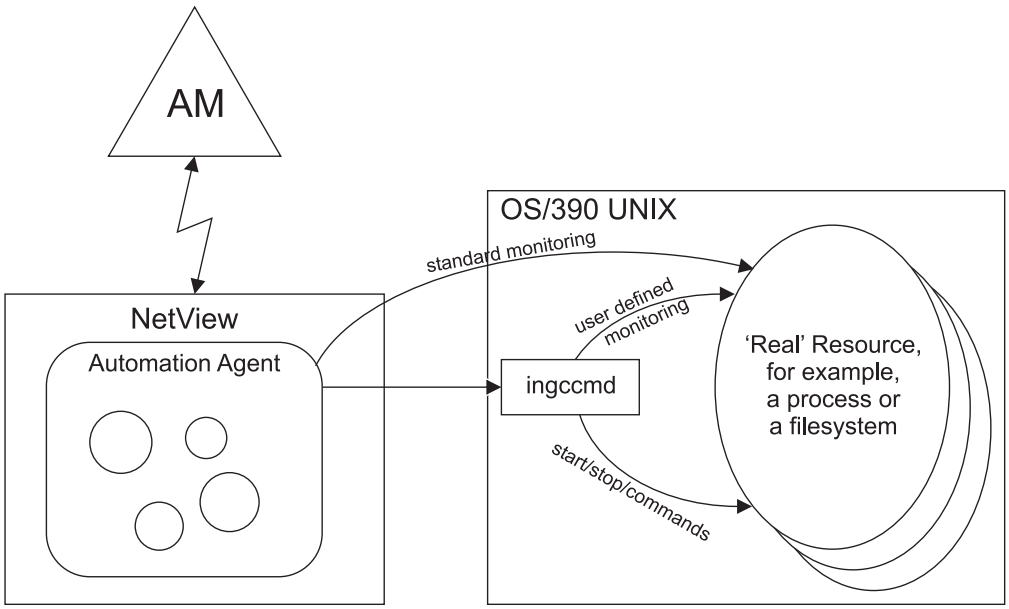


Figure 3. Automation of OS/390 UNIX System Services

Installing and Setting Up OS/390 UNIX Automation

Perform the following steps to install the OS/390 UNIX automation enhancements:

1. Copy the SA OS/390 UNIX System Services command server (ingccmd) from the MVS data set HLQ.SINGMOD1 to the OS/390 UNIX hierarchical file system (HFS). (See "Copying the SA OS/390 Command Server to the UNIX File System" for details.)
2. Define one or more UNIX segments, depending on the OPERSEC option of NetView. (See "Defining the UNIX Segments (OMVS)" on page 31 for details.)
3. Customize NetView. (See "Customizing NetView" on page 33 for details.)
4. If UNIX syslogd messages should be trapped, define an entry in the syslogd configuration file, in order to forward the messages to the MVS system log. (See "Trapping UNIX syslogd Messages" on page 46 for details.)
5. Restart NetView.
6. Refresh the configuration if you want to use the newly defined OS/390 UNIX resources.

Copying the SA OS/390 Command Server to the UNIX File System

A sample job (INGUSCPY) to copy the required file to the UNIX HFS is provided in &HLQ.SINGSAMP. Before submitting this job you have to modify it according to the instructions that are included in the job.

INGUSCPY copies the UNIX program required by SA OS/390 to the UNIX file system. The user submitting the job must have the appropriate authority to access and create the specified directory in the HFS and to set the extended attributes of the file (set program controlled bit 'on' and share address space bit 'off'). The program controlled bit is only needed if BPX.DAEMON RACF facility class is used. Otherwise the associated extended attribute (PGM) in the job can be deleted.

You can check the attributes with the `ls` command and the `-E` option as shown in the following example:

```
$ ls -E
-rwxr-xr-x  -p--  1 CAMP DE#03243  77824 Jan 31 08:36 ingccmd
```

If you copy the file later to the HFS, the extended attributes will be reset to their default value (--s-). In this case, an operator must set the extended attributes manually to the correct value. This is done by issuing the following command:

```
$ extattr +p -s ingccmd
```

Defining the UNIX Segments (OMVS)

Depending on the operator security definition of NetView, one or more UNIX segments must be defined. These OMVS segments can have a root UID (0) or a non-root UID. To run non-root requires more setup.

When using OPERSEC=MINIMAL, NETVPW, or SAFPW, one OMVS segment must be defined. This is the segment for the user ID running NetView.

When using OPERSEC=SAFCHECK, or SAFDEF (user level security), the following operator IDs need a UNIX segment:

- AUTWRK01-NN
- RPCOPER
- MONOPER
- AUTRPC
- AUTO1
- AUTSYS (backup task for AUTRPC and AUTO1)
- AUTBASE (backup task for AUTRPC and AUTO1)
- all tasks receiving actions from the MAT for UNIX resources, usually these are the work operators

Using the OMVS Segment with Root UID

This is the easiest way for setting up the OS/390 UNIX segment. To give it a UID of 0 (root user), enables this user to operate without restrictions. This segment must also be permitted to the RACF facility class BPX.DAEMON (if defined).

Note: Each user who can change NetView CGlobals may be able to issue UNIX System Services commands under a root user ID.

Using the OMVS Segment with Non-Root UID

If you want to reduce the number of UID 0 users, it is possible to define a setup without UID 0 with some restrictions.

If you are using a setup with non-root UID, the OMVS segment must be defined in the following way:

Monitoring:

- For process monitoring:

  Define read access to `SUPERUSER.PROCESS.GETPSENT`

  This allows a user ID to see all processes. If the user ID performing the monitoring is not allowed to check all processes, the automation may assume that the start was not successful and restarts the application. This will result in many instances.
- For file or file-system monitoring:

  Define read access to `SUPERUSER.FILESYS`

This allows a user ID to get access to all files in the UNIX file system. If the user ID performing the monitoring is not allowed to check all files, the automation may assume that the resource is unavailable.

- Give access to any resource that user-written monitoring routines may use
- For user-defined monitoring, see "Command Execution (INGUSS)" below. (User defined monitoring is performed with the command INGUSS.)

Command Execution (INGUSS):

- Give the OMVS segment the ability to switch to any user ID associated with OS/390 UNIX resources (access to `BPX.SRV.userid` or `BPX.SUPERUSER` to start root programs).
- Depending on your security environment the OMVS segment may need access to `BPX.DAEMON`.
- The OMVS segment must be authorized to perform all the commands that are specified in the customization dialogs. For an overview of authorizations for non-root users, refer to the chapter that explains UNIXPRIV class profiles in *z/OS UNIX System Services Planning*.

Restrictions for Non-Root UID Setup: There is an MVS identity and an OS/390 UNIX identity. Without a UID 0 you cannot switch the MVS identity. If a user needs access to certain MVS data sets, you may not start the application with INGUSS. You may have trouble when automating OS/390 UNIX resources that require a UID of 0 (for example, the inetd). The OMVS segments without UID 0 are normally not able to switch to a root user in order to perform actions. SA OS/390 standard monitoring will work. For example, if you allow the OMVS segment to switch to UID 0 (by defining read access to BPX.SUPERUSER), you could also assign it a UID of 0.

Creating an OMVS Segment by Submitting a Job

Creating OMVS segments can be done by submitting a job, as shown in Figure 4.

The NOPASSWORD option prevents unauthorized logins.

This OMVS segment must be authorized to set the jobname (read access to `BPX.JOBNAME`). Otherwise, the started address spaces have the same jobname as NetView. When the jobname can be set, the newly created address space has the jobname INGCUNIX.

```
//*
//ADDUSER  EXEC PGM=IKJEFT01
//*
//SYSTSPRT DD SYSOUT=*
//SYSLBC   DD  DSN=SYS1.BRODCAST,DISP=SHR
//SYSTSIN  DD *
  ADDUSER STCUSER +
          NOPASSWORD+
          UACC(NONE) DFLTGRP(AUTGRP) +
          OMVS(UID(0000000) HOME('/')  PROGRAM('/bin/sh')) +
//*
//COUSERS  EXEC PGM=IKJEFT01
//*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
  CO      STCUSER GROUP(USERS) AUTH(USE)
//*
```

Figure 4. Job Example of Creating an OMVS Segment

BPX.JOBNAME

If the started UNIX processes are to have a user-defined MVS jobname (specified with the JOBNAME parameter of the INGUSS command), the target user IDs that are issuing the commands must have at least read access to RACF facility class BPX.JOBNAME. Otherwise, a jobname will be assigned by the operating system. The target user ID is the user that this resource is assigned to in the customization dialog panel OS/390 UNIX Control Specification (see Figure 8 on page 36).

Customizing NetView

AOFUSSWAIT in AOFEXDEF

SA OS/390 introduces a new common global variable (AOFUSSWAIT) that you can set in your startup exit to change the way SA OS/390 behaves. This variable should be set only once for an SA OS/390 system. You can enable or disable advanced automation options by changing the settings of the global variables in your initialization defaults exit (AOFEXDEF).

AOFUSSWAIT is the time SA OS/390 waits for the completion of a user-specified OS/390 UNIX monitoring routine (defined in the OS/390 UNIX Control Specification panel) until it gets a timeout. When the timeout occurs, SA OS/390 does no longer wait for a response from the monitoring routine and sends a SIGKILL to the monitoring routine.

Add the following parameters to the AOFEXDEF exit to assign a value to the CGLOBAL AOFUSSWAIT:

```
aofusswait = 20
GLOBALV PUTC AOFUSSWAIT
```

If this value is not specified it defaults to 10 seconds.

Adding AOFUXMON and AOFRSUSS to Resident Clist List

For performance reasons the following clists should be added to the list of resident clists:

- AOFUXMON (the OS/390 UNIX monitoring routine)
- AOFRSUSS (for the INGUSS API)

Restarting NetView

In order to apply these changes, you have to restart NetView.

Customization of OS/390 UNIX Resources

The new OS/390 UNIX resources are introduced to SA OS/390 by defining them in the SA OS/390 customization dialogs.

The customization dialogs now support the application type USS. If USS is selected, you can enter the OS/390 UNIX-specific data such as UNIX user ID, command or path, filename, or monitored port. Choose one of these fields to enter the data.

The start and stop definitions can be varied between MVS and OS/390 UNIX commands. For example, to stop an application you can issue a UNIX kill command first and (if this was not successful) you can perform an MVS cancel later.

Definitions for Automation Setup

The HFS path, where the program shipped with SA OS/390 is located, must be defined in the SA OS/390 setup panel, as shown in Figure 5. This has to be the same path that was used as thedestination in the sample job INGUSCPY. When user-defined UNIX monitoring is used and no absolute path is specified for the monitoring routine, SA OS/390 tries to start the user-defined monitoring routine in this directory.

```
   COMMANDS  HELP
-------------------------------------------------------------------------------
AOFPIES0                        Environment Setup
Command ===> ____________________________________________________________

Entry Type : System               PolicyDB Name   : GFG_TESTPDB
Entry Name : GFG1                 Enterprise Name : GFG_TESTPDB

Enter the following information:

Primary JES . . . . . . . ________  Primary JES2/JES3 subsystem name
System Monitor Time . . . ______    Time between SYSTEM monitoring
                                    cycles (hh:mm or NONE)
Gateway Monitor Time  . . ______    Time between GATEWAY monitoring
                                    cycles (hh:mm or NONE)
Monitor Option  . . . . . ________  Default routine used to monitor
                                    application status
Message Table . . . . . . ____________________________________________
                                    Netview message automation table members
MVS SYSNAME             : GFG1      The MVS SYSID in SYS1.PARMLIB
SDF Root Name . . . . . . GFG1      Root of this system's SDF tree
Exit name(s)  . . . . . . ____________________________________________
                                    Environment setup user exit names
Path. . . . . . . . . . .
/u/camp/sandbox
_____________________________________________________________________
_____________________________________________________________________
                                    SA UNIX installation path
```

Figure 5. Environment Setup Panel

Definitions for OS/390 UNIX Resources

To define a new application entry (APL, class, or instance), specify the application type USS on the Define New Entry panel, as shown in Figure 6 on page 35.

```
  COMMANDS  HELP
------------------------------------------------------------------------------
AOFGLN00                        Define New Entry
Command ===> ____________________________________________________________

To define a new entry, specify the following information:
   Type. . . . . . . . . . Application
   Application Name. . . . ____________________
   Subsystem Name. . . . . ____________
------------------------------------------------------------------------------
                                                                    More:     +
   Object Type . . . . . . INSTANCE   CLASS     INSTANCE
   Application Type . . .  USS        STANDARD JES2 JES3 CICS IMS DB2 OPC USS
                                      (Only the value STANDARD can be changed)
                                      (once, all others cannot be changed)
                                      (after the application has been created)
   Subtype . . . . . . . . ________   (Only for type CICS, IMS, OPC or DB2)
   Clone Job Name. . . . . NO         YES       NO
   Job Name. . . . . . . . ____________________________
   Scheduling Subsystem. . ________   MSTR, JES Subsystem or blank
   JCL Procedure Name. . . ________   (Proc used with JOBNAME=)
   MVS Automatic Restart   ________
     Management Element
     Name. . . . . . . . . ____________________________________
                                      (Only if the application uses)
                                      (MVS Automatic Restart Management)
   WLM Resource Name . . . ____________________________________

   Short Description . . .
------------------------------------------------------------------------------
```

Figure 6. Defining Application Type USS

When choosing the application type USS, the new option USS Control is displayed on the Policy Selection panel, as shown in Figure 7.

```
   ACTIONS  HELP
------------------------------------------------------------------------------
AOFGEPOL                        Policy Selection            Row 1 to 22 of 22
Command ===> ____________________________________________  SCROLL===> PAGE

Entry Type : Application          PolicyDB Name   : GFG_TESTPDB
Entry Name : STEFANTEST1          Enterprise Name : GFG_TESTPDB

Action      Policy Name          Policy Description
__________  DESCRIPTION          Enter description
__________  LINK TO CLASS        Link instance to class
__________  APPLICATION INFO     Enter and display Application information
__________  AUTOMATION INFO      Define Application automation information
__________  AUTOMATION FLAGS     Define Application automation flags
__________  TRIGGER              Select trigger
__________  SERVICE PERIOD       Select service period
__________  RELATIONSHIPS        Define relationships
__________  MESSAGES/USER DATA   Define Application messages and user data
__________  STARTUP              Define startup procedures
__________  SHUTDOWN             Define shutdown procedures
__________  THRESHOLDS           Define error thresholds
__________  MINOR RESOURCE FLAGS Define Application sub-component flags
__________  SYSTEM ASSOCIATION   Define primary and secondary associations
            -------------------- -----RESOURCES-------------------------------
__________  GENERATED RESOURCES  List resources generated for this entry
__________  MEMBER OF            List resources where this entry is a member
            -------------------- -----USS SPECIFIC POLICY---------------------
__________  USS CONTROL          Define USS Control specifications
            -------------------- ---------------------------------------------
__________  WHERE USED           List ApplicationGroups linked to this entry
__________  COPY                 Copy data from an existing entry
```

Figure 7. Defining UNIX System Services Control Specifications

When selecting USS Control on the Policy Selection panel, you can enter the data for the new OS/390 UNIX resource. For a class, only the user ID and the OS/390 UNIX monitoring routine can be specified on this panel (see Figure 8). All other definitions (for example, start/stop, dependencies) can be entered as usual.

```
COMMANDS  HELP
------------------------------------------------------------------------------
AOFGUSS                    USS Control Specification
Command ===> ____________________________________________________________

Entry Type : Application           PolicyDB Name   : GFG_TESTPDB
Entry Name : USSCLASS              Enterprise Name : GFG_TESTPDB

Subsystem name: USSCLASS

User ID . . . . . ________

Monitoring Command. . .
 ____________________________________________________________________________
 ____________________________________________________________________________

Further definitions have to be done for the instance objects.
```

Figure 8. OS/390 UNIX Control Specification Panel for Type CLASS

For object type INSTANCE you can define whether this resource is a process, a TCP port, or a file or file system, as shown in Figure 9.

```
   COMMANDS  HELP
 ------------------------------------------------------------------------------
 AOFGUSS                    USS Control Specification
 Command ===> ____________________________________________________________

 Entry Type : Application           PolicyDB Name   : GFG_TESTPDB
 Entry Name : STEFANTEST1           Enterprise Name : GFG_TESTPDB

 Subsystem name: STEFANTEST1

User ID . . . . . camp___

Monitoring Command. . .
 ____________________________________________________________________________
 ____________________________________________________________________________

 Enter or update one of the following fields:

 Command/Path. . .
/u/camp/usstest/usstest____________________________________________________
 ____________________________________________________________________________

 File Name . . . .
 ____________________________________________________________________________
 ____________________________________________________________________________

 Monitored Port. . _____
```

Figure 9. OS/390 UNIX Control Specification Panel for Type INSTANCE

If the field *Monitoring Command* is left blank, SA OS/390 internal monitoring is invoked by AOFUXMON. If a command is specified, this command is issued instead of the SA OS/390 internal monitoring. The first part of the command must be an executable program (for example, this can be a shell script, a compiled C program, or a REXX program). This is the UNIX monitoring routine. You can also add other options for this program. In the field *Monitoring Command* you can

define the global variables &SUBSPATH, &SUBSPORT, &SUBSFILE, &SUBSJOB, &SUBSAPPL, and &SUBSUSER, as well as system clone variables.

When the executable file does not begin with a "/" it must reside in the same directory as the SA OS/390 supplied OS/390 UNIX routine ingccmd. Otherwise the name specified is considered to be an absolute path identifier.

The UNIX monitoring routine must have an exit value. It can be one of the following:

0 Resource is available

4 Resource is starting

8 Resource is unavailable

12 Error occurred

If the user-specified monitoring routine loops, it will receive a SIGKILL after the AOFUSSWAIT time (defined in AOFEXDEF).

Hint:

It is possible to write a message from this UNIX monitoring routine to the MVS system log, in order to trigger an action or perform a status change through the Message Automation Table (MAT).

Note: There are two monitoring routines:

- AOFUXMON, which is called by SA OS/390 for UNIX System Services resources. (This must always be specified).
- A program in the HFS which is entered in the OS/390 UNIX Control Specification panel, and is called by AOFUXMON. This means that if you specify this monitoring command, you also have to specify AOFUXMON in the Automation Application Definition panel, as shown in Figure 10 on page 38.

```
  COMMANDS  HELP
------------------------------------------------------------------------------
AOFPISS1               Application Automation Definition
Command ===> ____________________________________________________________

Entry Type : Application          PolicyDB Name   : GFG_TESTPDB
Entry Name : USS1                 Enterprise Name : GFG_TESTPDB

Subsystem  : USS1
Description:
Job Name   : USS1
                                                              More:     +

Job Type . . . . . NONMVS     Job properties (MVS NONMVS TRANSIENT)
Transient Rerun  . ___        Transient Jobtype can be rerun (YES NO)
Command Prefix   . ________   Console command character(s)
Message Prefix . . ____________________________________________________
                              Enter one or more prefixes (above)

Sysname  . . . . . ________   System name used by the application


Start on IPL . . . ____       Start with Netview init (YES NO NONE blank)
Start on Recycle . ____       Start with Sys-Ops recycle (YES NO NONE blank)
Start Timeout  . . ________   Time allowed to reach "UP" status (hh:mm:ss)

Monitor Routine. . AOFUXMON   Routine used for monitoring (name NONE)
Periodic Interval. _____      Periodic monitoring interval (hh:mm NONE)

Restart Option . . _________  Restart Circumstances (ALWAYS ABENDONLY NEVER)
External Startup . _______    External Startup (INITIAL ALWAYS NEVER blank)
External Shutdown. _______    External Shutdown (FINAL ALWAYS NEVER blank)

Shut Delay . . . . ________   Time between attempts to shutdown (hh:mm:ss)
Term Delay . . . . ________   Time for termination cleanup (hh:mm:ss)
```

Figure 10. Application Automation Definition Panel

The monitoring routine AOFUXMON must be entered in the Application Automation Definition panel. If you do not specify this routine, the default monitoring routine (usually AOFAJMON) would be called which is not sufficient for OS/390 UNIX resources.

The job type can be either MVS or NONMVS.

MVS Is only used for resources which represent a process with a unique jobname. For these resources SA OS/390 accepts the following messages for status changes:

- IEF403I Job started
- IEF404I Job ended
- IEF450I Job abended

If no start command is specified, the default MVS start method (`s <JOBNAME>`) is used.

NONMVS
SA OS/390 ignores the messages listed above for status changes. This is neccessary if the jobname is not unique.

For OS/390 UNIX resources the timeout and delay times have the following meaning:

Start Timeout
The start timeout interval begins when SA OS/390 issues a start command for an application. After the start timeout the monitoring method is triggered. When the monitor detects the resource as available, the agent

status is set to 'active'. After another start timeout interval and successful monitoring, the ACTIVMSG generic routine is triggered which sets the agent status to 'up'. The default value for Start Timeout is 2 minutes.

Shut Delay
The time between the different shutdown passes

Term Delay
The time to allow cleanup after the termination event

Automated Resources

Process Monitoring: No UNIX process identifiers (PIDs) can be monitored. The monitoring routine needs the start command and the user ID the process belongs to. This information can be obtained with the UNIX command `ps`. In the following example all processes belonging to user CAMP are displayed:

```
CAMP:/u/camp/ingcmd>ps -e -o comm
COMMAND
/bin/sh
/usr/sbin/rlogind2
/bin/ps
/bin/sh
/usr/sbin/rlogind2
CAMP:/u/camp/ingcmd>
```

That means that automation could not distinguish between the two processes started by `/usr/sbin/rlogind2`. Processes started by identical commands must have different user IDs.

If it is necessary to automate processes running multiple instances, a user could use softlinks to distinguish between the different processes. For example, the process:

```
/u/camp/usstest/testme
```

should be started more than once. In this case, create some softlinks:

```
CAMP:/u/camp/usstest> ln -s testme test1
CAMP:/u/camp/usstest> ln -s testme test2
```

This results in:

```
CAMP:/u/camp/tt>ls -al
total 216
drwxrwxr-x   2 CAMP     DE#03243    8192 Jan 24 16:24 .
drwxr-xr-x  19 CAMP     DE#03243    8192 Jan 24 16:23 ..
lrwxrwxrwx   1 CAMP     DE#03243       6 Jan 24 16:24 test1 -> testme
lrwxrwxrwx   1 CAMP     DE#03243       6 Jan 24 16:24 test2 -> testme
-rwxrwxr-x   1 CAMP     DE#03243   94208 Jan 24 16:23 testme
```

These three programs (being the same ″real″ program) can be automated with the three different start commands test1, test2, and testme. These links may be created as a prestart command and deleted as a shutfinal command.

Note: Only the command is used, not the parameters that were used to start the program. This is because a program may be started by SA OS/390 with different startup parameters, depending on what the automation manager told the automation agent to do. In this case, the only constant value is the command, not the parameters.

TCP Port Monitoring: Exactly one TCP port number can be entered for one resource. SA OS/390 monitors the local host as returned by the function

gethostid(). When this port has a state of 'listening', this resource is considered to be 'available' in terms of SA OS/390. All other states of the port will map to 'unavailable'.

File or File-System Monitoring: The existence of a file (belonging to a certain user) is verified. Many applications create files at startup and delete these files when terminating normally. If more than one file should be monitored, this can be modeled as an application group (APG) in the automation manager.

This monitoring can be used to determine if a certain file system is mounted. The start command for this resource would be a UNIX 'mount' command, the stop command a UNIX 'umount'.

Start and Stop Definitions (INGUSS Command)

If the resource is to be controlled by traditional MVS commands, this could be done in the same way as for all other MVS applications. Issuing commands in the OS/390 UNIX environment is done by specifying the INGUSS command at the start or stop definitions.

Note: INGUSS can only be used if the primary JES is available. Therefore, OS/390 UNIX resources using INGUSS need a HASPARENT dependency to JES. Most OS/390 UNIX applications have this dependency. If you want to issue prestart commands, an additional PREPAVAILABLE dependency is necessary.

OS/390 UNIX and MVS commands can be mixed in different shutdown passes.

The syntax of the OS/390 UNIX command is:

where

JOBNAME=*jobname*
: *jobname* is the MVS jobname used for the newly created address space which runs the specified command. If you do not specify a jobname, INGCUNIX is the default.

UNIX_command
: This is the OS/390 UNIX command that is issued from the user ID of the resource this definition belongs to. It is not possible to issue commands for other user IDs. It can be any OS/390 UNIX command or the name of a shell script (fully qualified).

You can use three additional variables to obtain static data of the resource:

&SUBSPATH
: The path statement of the resource (must be a process).

&SUBSFILE
: The filename of the resource (must be a file)

&SUBSPORT
: The port number of the resource (must be a port)

The content of these variables is defined on the USS Control Specification panel (see Figure 9 on page 36).

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.

Usage of &SUBSUSSJOB and &SUBSASID: &SUBSUSSJOB and &SUBSASID can be used in the stop definitions for type process only. &SUBSUSSJOB is the jobname assigned to the process. &SUBSASID is the address space ID of the address space the process runs in. This information is refreshed with each monitoring cycle, so if a process forks and gets a new jobname (normally appending a digit at the end of the original jobname) SA OS/390 will detect the new jobname after the next scheduled monitoring.

When the resource becomes inactive, the values of &SUBSUSSJOB and &SUBSASID are cleared.

For job type MVS, &SUBSJOB is adjusted to the value of &SUBSUSSJOB when the resource is active. &SUBSJOB is reset to its value specified in the ACF when the resource becomes inactive.

This functionality can be used with SA OS/390 internal monitoring only, which sets the appropriate values. If a user-specified OS/390 UNIX monitoring routine (in the USS Control Specifications panel) is used, it does not work.

Getting the User Environment when Starting a Program: When issuing a command SA OS/390 switches to the users 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 you want the started program to get the whole environment of the user as though this user was logged on, you must use a login shell as the start command.

If you want to start the inetd through a login shell, issue the command:

```
INGUSS JOBNAME=INETD /bin/sh -L -c '/usr/sbin/inetd /etc/inetd.conf'
```

where

JOBNAME=INETD
: This is optional. It assigns the MVS jobname 'INETD' to the started process, described below.

/bin/sh
: The shell

-L Option for login shell

-c 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

Jobname of the Started Address Spaces: As mentioned before, all UNIX commands are executed by the program ingccmd, provided with SA OS/390. When invoked, this program gets the jobname INGCUNIX. This means that all programs started by this mechanism inherit this jobname unless another one is specified with the JOBNAME parameter of the INGUSS command.

Recommendation:

When using INGUSS to start applications, IBM recommends that you use the JOBNAME parameter in order to get a unique jobname, for example:

```
INGUSS JOBNAME=&SUBSJOB UNIX_start_command
```

Otherwise, all applications started by SA OS/390 without this parameter will have the same jobname INGCUNIX (if the application itself does not change the jobname).

If the jobname is not unique, specify job type NONMVS.

Command Examples:

Start Command for a Process: To start a process with the command and jobname specified in the customization dialogs, enter INGUSS JOBNAME=&SUBSJOB &SUBSPATH on the Startup Command Processing panel, as shown in Figure 11.

```
   COMMANDS  HELP
------------------------------------------------------------------------------
                           Startup Command Processing         Row 1 to 1 of 21
Command ===> ____________________________________________________ SCROLL===> PAGE

Entry Type : Application           PolicyDB Name   : USS_AO
Entry Name : USS_TEST_APPLICATION  Enterprise Name : USS_AO

Subsystem       : USS_TST_APL
Startup Phase   : STARTUP

Schedule Subsys . ________   MSTR, JES Subsystem or blank
JCL Proc Name . . ________   (Proc used with JOBNAME=)
Parameters. . . . _____________________________________________________
                           Enter Subsystem Startup Parameters(above)

Type         Automated Function/'*'
Command text
_____ _______
INGUSS JOBNAME=&SUBSJOB &SUBSPATH
_____________________________________________________________________________
```

Figure 11. Startup Definition for a Process

Only the command that was used to start an application or a process can be monitored. If the same program should be started multiple times, a softlink as prestart command could be used to distinguish the processes.

Use a Softlink to Distinguish Processes Running the Same Executable as Prestart Command: The following panel creates a softlink for &SUBSPATH (the path parameter of the resource issuing the command, for example, /u/user1/uss1) and links to the file /u/user1/usstest.

```
Type         Automated Function/'*'
 Command text
              *
 INGUSS /bin/ln -s /u/user1/usstest &SUBSPATH
```

Figure 12. Creating a Softlink

When looking at the HFS, this results in:

```
USER1:/u/user1>ls -l
total 408
lrwxrwxrwx   1 USER1     DE#03243       7 Feb 13 12:44 uss1 -> usstest
-rwxrwxr-x   1 USER1     DE#03243  163840 Jan 29 14:55 usstest
```

Stop Commands for a Process: An OS/390 UNIX process may be stopped in different ways (escalation passes). For example, you can first use the OS/390 UNIX `kill` command, if that does not work use OS/390 UNIX `kill -9`, and finally enter an `MVS cancel` command.

Enter the definitions for this example as shown in Figure 13.

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                           Shutdown Command Processing        Row 1 to 2 of 23
Command ===>                                                   SCROLL===> PAGE

Entry Type : Application           PolicyDB Name   : USS_AO
Entry Name : USS_TEST_APPLICATION  Enterprise Name : USS_AO

Subsystem     : USS_TST_APL
Shutdown Phase: NORM

Enter commands to be executed when the selected shutdown phase is invoked
for this subsystem.

Pass         Automated Function/'*'
Command Text
1
INGUSS /bin/kill %PID%


3
INGUSS /bin/kill -9 %PID%


4
MVS C &SUBSUSSJOB,A=&SUBSASID
```

Figure 13. Stop Definitions for a Process

%PID% is replaced at run time by the real PID of the process.

Stop Command for a File: A stop command for a file may be deleting the file. The filename entered in the customization dialogs can be found in &SUBSFILE.

```
Pass         Automated Function/'*'
 Command Text
 1
 INGUSS /bin/rm &SUBSFILE
```

Figure 14. Delete a File

Debugging

Debugging can be activated for OS/390 UNIX monitoring and command execution on the AOCTRACE panel. The clist for monitoring is AOFUXMON and for command execution AOFRSUSS.

The debugging messages will be written to the netlog and to the OS/390 UNIX system log (syslogd).

Example: inetd

The inetd is the UNIX internet daemon. It allows you to invoke several others and it should be started at IPL time (normally through `/etc/rc`). It then listens for connections to certain internet sockets. Its configuration file is `/etc/inetd.conf`.

The following is a sample inetd configuration file:

```
login    stream  tcp  nowait  OMVSKERN  /usr/sbin/rlogind rlogind -m
exec     stream  tcp  nowait  OMVSKERN  /usr/sbin/orexecd orexecd -d
otelnet  stream  tcp  nowait  OMVSKERN  /usr/sbin/otelnetd otelnetd -k -t
daytime  stream  tcp  nowait  OMVSKERN  internal
time     stream  tcp  nowait  OMVSKERN  internal
netbios-ssn stream tcp nowait OMVSKERN  /local/samba/bin/smbd smbd
```

When a service request is detected at one of its sockets, it decides what service the socket corresponds to and invokes a program to service the request. Then it normally continues to listen to the socket that the last request came in at (see Figure 15).

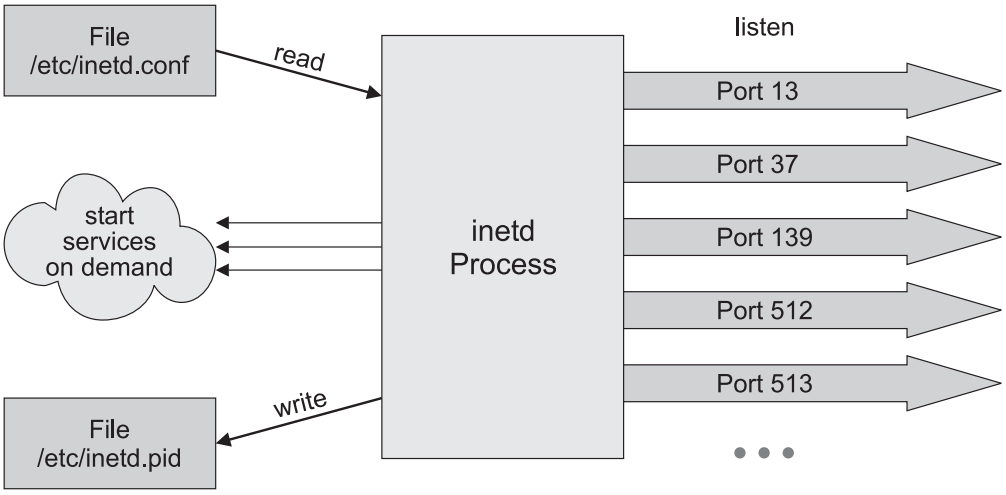


Figure 15. inetd Structure

The inetd started with the configuration file above will listen to the following sockets:

```
CAMP:/etc>netstat -a | grep INET
INETD1   00006B80 0.0.0.0..13          0.0.0.0..0          Listen
INETD1   00006B7D 0.0.0.0..513         0.0.0.0..0          Listen
INETD1   00006B7E 0.0.0.0..512         0.0.0.0..0          Listen
INETD1   00006B7F 0.0.0.0..623         0.0.0.0..0          Listen
INETD1   00006B82 0.0.0.0..139         0.0.0.0..0          Listen
INETD1   00006B81 0.0.0.0..37          0.0.0.0..0          Listen
```

Whereas the services and the real port numbers correspond according to `/etc/services`:

```
daytime           13/tcp           #Daytime
time              37/tcp timserver #Time
netbios-ssn      139/tcp           #NETBIOS Session Service
exec             512/tcp           #remote process execution;
login            513/tcp           #remote login a la telnet;
otelnet          623/tcp           #OE telnet
```

The UNIX internet daemon (inetd) can be defined in the customization dialogs, for example:

```
Application Name: INETD/APL   Application Type: USS
Command/Path: /usr/sbin/inetd   User ID: OMVSKERN
Port: -      File:


Application Name: INETFILE/APL  Application Type: USS
Command/Path:      User ID: OMVSKERN
Port: -      File: /tmp/inetd.pid


Application Name: INETPORT/APL  Application Type: USS
Command/Path:      User ID: OMVSKERN
Port: 513      File:
```

Define a basic group containing all resources with relationships that indicate that:

- The file is created by the inetd process and can never be started or created directly by SA OS/390.
- The inetd process listening to the port can never be started or created directly by SA OS/390.

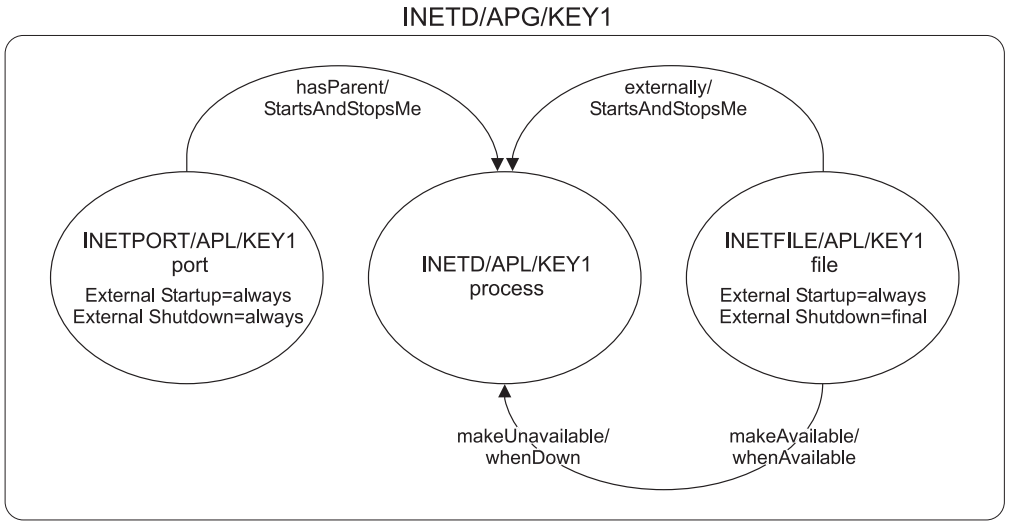


Figure 16. Dependency Graphic

The example above recognizes the inetd (modeled as a group) as up and running when the process `/usr/sbin/inetd` started by user OMVSKERN shows up, the file `/tmp/inetd.pid` exists and port 513 is in status 'listen' (inetd will listen to this port for incoming login requests).

You can only choose a port that is defined in `inetd/conf`

Start definition for INETFILE/APL
None

Start definition for INETPORT/APL:
None

Start definition for INETD/APL:

CMD: `INGUSS JOBNAME=&SUBSJOB &SUBSPATH /etc/inetd.conf`

(&SUBSPATH is substituted at run time by the parameter command/path.)

Stop definitions for INETFILE/APL:

CMD: `INGUSS /bin/rm &SUBSFILE`

(This will remove the file if not yet removed by the inetd process.)

Stop definition for INETPORT/APL:
None

Stop definitions for INETD/APL:

```
CMD: INGUSS /bin/kill %PID%
CMD: INGUSS /bin/kill -9 %PID%
CMD: MVS C &SUBSUSSJOB,A=&SUBSASID
```

%PID% will be replaced by the OS/390 UNIX command routine with the real PID which matches the parameters command/path and user ID. In the following example this is 33554821.

```
CAMP:/u/camp/ingcmd>ps -e -o pid,comm -u OMVSKERN
       PID COMMAND
  33554481 /bin/sh
  50331698 /usr/sbin/rlogind2
  33554486 /usr/lpp/netview/bin/cnmeunix
  67108927 /bin/sh
  83886176 /bin/ps
  33554821 /usr/sbin/inetd
  83886472 FTPD
  67109276 /bin/sh
  16777629 /usr/sbin/rlogind2
  33554924 HSAPYTCP
```

Hints and Tips

Trapping UNIX syslogd Messages

To trap UNIX syslogd messages, an entry in the syslogd configuration file `/etc/syslog.conf` must be added in order to forward the messages to the MVS system log. Thus, messages can be processed by the Message Automation Table (MAT).

To forward all messages to the MVS log add the following entry:

```
*.* /dev/console
```

To send special messages to the MVS log only, follow the syslog message naming guidelines (for example, for warning messages use *.warn). `/dev/console` can be used as an ordinary file to write to.

The UNIX messages have the MVS message ID BPXF024I and are multiline messages.

Figure 17 shows an example of a UNIX message:

```
M 13:45:21.34 STC03602 00000090 BPXF024I (CAMP) Feb 13 13:45:21 BOEKEY1 syslogtest 67109100 : This is
S                               498
D                  498 00000090 a test message
```

Figure 17. Example of a UNIX Message

Hints and Tips

Chapter 5. How to Enable Enhanced Parallel Sysplex Automation

This chapter describes the enhancements to Parallel Sysplex automation and how to use the SA OS/390 customization dialogs to enable them.

Note: If you use a host code page other than 037, the hexadecimal representation of the at sign (@) can be different. Use the letter represented by the hex code X'7C' for the at sign.

Enhanced Sysplex Functions

This section gives an overview of the SA OS/390 sysplex functions that can now be automated and the new command-driven actions.

Managing Couple Data Sets

Couple data sets (CDSs) contain control information about the sysplex and its resources, and are of crucial importance for the functioning of a Parallel Sysplex. Particularly important are the SYSPLEX couple data set, which contains information about the systems and the communication structure (XCF groups) of the sysplex, and the CFRM couple data set, which specifies its coupling facilities (CFs) and structures (see "Managing Coupling Facilities" on page 52). Every MVS system in a Parallel Sysplex must have access to these CDSs, and to those of all other implemented sysplex functions, such as SFM and Application Response Measurement (ARM).

If a member system cannot access a CDS, the corresponding sysplex function is impacted, and in some cases the sysplex will go down. It is therefore recommended that you define two CDSs to XCF for every CDS type required for the implementation of the sysplex. One of these, the *primary* CDS, is the one that is actually used. The other, which is called the *alternate* CDS, serves as a backup copy. The two CDSs contain the same data. Whenever the primary CDS changes, XCF updates the alternate CDS accordingly. If an alternate CDS is available for a certain type, XCF automatically switches to this alternate CDS whenever a member can no longer access the primary CDS.

All CDSs except the sysplex couple data set contain one or more user-defined configurations, called *policies*. For each CDS type, only one policy can be active. However, it is possible to switch the active policy at run time. Refer to "INGPLEX CDS" on page 50 for further information.

SA OS/390 offers two functions for easier CDS management:

- Automated creation and recovery of alternate couple data sets for continuous availability
- INGPLEX CDS, which simplifies management of couple data sets

The following describes the enhancements of SA OS/390 for managing couple data sets.

Ensuring Continuous Availability of Couple Data Sets

When an alternate CDS exists for a given CDS type and the current primary CDS fails, XCF makes this alternate the primary CDS. After this switch, however, an

alternate CDS no longer exists, and if the current primary CDS also fails, the problems that were to be avoided by the creation of an alternate occur again. To avoid this single-point-of-failure situation, SA OS/390 provides a recovery mechanism that tries to ensure that an alternate CDS is always available for every CDS type used.

SA OS/390 creates a new alternate CDS in the following two situations:

- During initialization, SA OS/390 checks that an alternate CDS is specified for every primary CDS. If there is a primary CDS for which no alternate CDS exists, SA OS/390 automatically creates it.
- At run time, SA OS/390 ensures that a new alternate is created whenever the current alternate has been removed or switched to the primary one.

Customization

Recovery of alternate CDSs is initiated either by the CDS function of INGPLEX or in the background (for example, at initialization time). Background recovery can be switched on and off by using the SA OS/390 customization dialogs. Automatic re-creation with INGPLEX CDS is always enabled.

You must specify the spare volumes that SA OS/390 may use for creating missing alternate CDSs (using the policy item SYSPLEX from the ***Policy Selection*** panel for sysplex groups). This is also required for automatic creation with INGPLEX CDS. Every CDS type has its own pool of spare volumes. Note that if you do not define spare volumes for a CDS type, no recovery will be performed for this type. For details on the use of the customization dialogs, see "Enabling Continuous Availability of Couple Data Sets" on page 66.

You can control access to those functions of INGPLEX CDS that modify the sysplex configuration. Refer to Appendix A of *System Automation for OS/390 Planning and Installation* for details.

INGPLEX CDS

INGPLEX CDS displays information about all couple data sets, including details of the respective policies, and allows you to perform the following actions for every CDS type that is required for the implementation:

- Switch from the primary to the alternate CDS
- Define a new alternate CDS
- Change the active policy (if applicable)
- Automatically rebuild a structure after the activation of the CFRM policy

For the first two actions, INGPLEX CDS offers automatic creation of a new alternate CDS. You can also specify your own alternate CDS. For more information on INGPLEX CDS, see the online help.

Managing the System Logger

Terms and Concepts

The *system logger* provides a sysplex-wide logging facility. Applications that use the system logger write their log data into *log streams*. Within a Parallel Sysplex, these log streams are usually associated with a coupling facility structure. For further information about coupling facility structures, refer to "Managing Coupling Facilities" on page 52. By using a coupling facility log stream, members of a multisystem application can merge their logs even when residing on different systems.

When an application writes data to a log stream this data is stored at first temporarily in the associated structure (coupling facility log stream) or a local buffer (DASD-only log stream). From there, it is offloaded into a log stream data set which is automatically allocated by the system logger. When this log stream data set is full, the system logger allocates a second one, and so on.

The control information for the system logger, which includes a directory for the log stream data sets of every log stream, is contained in the LOGR couple data set. The total number of log stream data sets that can be allocated by the system logger is determined when the LOGR couple data set is formatted.

Two problems that can arise in connection with the log stream data sets are a shortage of directory space in the LOGR CDS and incorrect share options for the log stream data sets. SA OS/390 provides the following recovery actions for these problems:

- The primary and alternate LOGR CDSs are automatically resized if there is a directory shortage
- The operator is notified if the share options for log stream data sets are not defined correctly

Resizing the LOGR Couple Data Sets in Case of Directory Shortage

The LOGR CDS contains information about the log stream data sets used by the system logger. This information is stored in *directory extents*. Every directory extent record can hold information about up to 168 log stream data sets. The number of directory extents available in a LOGR CDS is specified when the CDS is formatted (`DSEXTENT` parameter). When all available directory extents are used up the system logger can no longer allocate new log stream data sets. This can cause considerable problems for applications that use the system logger.

With SA OS/390, you can avoid this situation. If you switch on logger recovery, SA OS/390 automatically reformats your primary and alternate LOGR CDS with an increased DSEXTENT parameter whenever the system reports a directory shortage.

Notifying the Operator of Incorrect Share Options

Note: This section applies to z/OS 1.2 and below.

If you wish to use the system logger, you must define share options for the log stream data sets. Merging data from several systems into one coupling facility log stream requires you to specify VSAM SHAREOPTIONS(3,3) for the log stream data sets. With other share options, especially (1,3), such a merge will fail. If you manage your DASD data sets with SMS (Storage Management Subsystem), a possible cause for incorrect share options is that the data class you use for the log stream data sets is also used for other purposes that require different share options.

SA OS/390 provides a control mechanism for VSAM share options. The share options are checked on a daily basis. If incorrect share options are detected, SA OS/390 notifies the operator.

Customization

Automation of system logger recovery is enabled through the SA OS/390 customization dialogs. For more details, see “Enabling System Log Failure Recovery” on page 67.

Managing Coupling Facilities

A *coupling facility* (CF) is a logical partition that provides storage for data exchange between components of an application that is distributed across different systems in a Parallel Sysplex. A Parallel Sysplex can contain more than one CF. The storage of a coupling facility is divided into areas that are called *structures*. You can imagine a structure as a special kind of data set. It is these structures, which are identified by their name, that are accessed for reading and writing by the application components.

The association between CFs and structures is dynamic. A structure that is used by an application need not be allocated at all (for example, when the application is not running), and can be allocated on different CFs at different points in time. For every structure, there exists a *preference list* that defines the CFs on which it may be allocated. The order of the CFs in that list determines which CF is selected when more than one member of the list satisfies all allocation requirements (for example, provides enough space).

The preference list, the space requirements, and other properties of the structures are defined in the active CFRM policy. This policy is contained in the CFRM couple data set. Refer to "Managing Couple Data Sets" on page 49 for further information.

XES allocates a structure that does not yet reside on any CF when an application component needs to be connected to it. Note that the application component only specifies the name of the structure that it wants to access. It is XES that decides on which CF the structure is allocated. This decision is influenced by the structure definition in the active CFRM policy. After the structure has been allocated, the requesting application component can access it, and further components of this application can require to connect to it. An application component that has access to an allocated structure is referred to as an *active connector* to this structure.

In the simplest case, XES deallocates a structure when all connected application components have disconnected from the structure. However, an application component can require that the structure or its own connection to the structure be *persistent*. When the *structure* is persistent it remains allocated even when the application component is no longer connected to it. When a *connection* is persistent the structure remains allocated after a failure of that connection. The application component in question remains a connector to the structure, although not an active one. It is now a *failed persistent* connector. In both cases, you can force the deallocation of the structure as soon as it no longer has active connectors.

Allocated structures can be *rebuilt*. Rebuilding is the process of reconstructing a structure on the same or another CF. A rebuild consists of three main steps. First, XES allocates the new structure instance. Then, the data of the old structure is reconstructed in the new structure. Finally, XES deallocates the old structure instance. Note that you cannot specify the target CF in your rebuild request. As with structure allocation, XES selects it from the preference list.

There are two methods for rebuild: user-managed and (from OS/390 2.8 onward) system-managed. With user-managed rebuild, the active connectors are responsible for reconstructing the data. With system-managed rebuild, XES transfers the data to the new structure instance. System-managed rebuild is thus also available for structures without active connectors. These structures can either themselves be persistent or have failed persistent connections.

When an application component connects to a structure, it specifies whether it allows the structure to be rebuilt through user-managed or system-managed rebuild. For structures with active connectors, both rebuild methods require that all active connectors allow the respective rebuild method.

You can also *duplex* structures. Duplexing means maintaining two instances of the same structure on different CFs at the same time. Duplexing serves to increase availability and usability of a structure.

Typical management tasks for CFs are removing a CF from the sysplex and reintegrating it again. These tasks have several steps that must be performed in a certain order and can be quite complex. To simplify these operations, SA OS/390 offers the INGCF command. INGCF has several functions, which serve to manipulate structures and the CFs themselves. These functions are briefly described in the following. For more information, see the online help.

Some functions deal with the sender paths of a coupling facility. They have the following limitations. First, at least one system in the sysplex that is running the automation must know the control unit id (CUID) of the coupling facility. If this is not the case, no missing sender paths can be resolved.

A missing sender path occurs when a coupling facility is deactivated prior to a system IPL (or reIPL) and then activated afterwards. The system that has been IPLed (or reIPLed) does not recognize the coupling facility. To determine the missing sender paths, the automation calls the HOM interface of HCD. Resolving the missing path information is only possible when either the complete network address is defined in HCD along with the processor id, or you provide the CPC synonym used by the automation as the processor id. However, it is recommended that you define both. If neither is defined, the system that misses the sender paths must run the automation.

INGCF DRAIN

INGCF DRAIN displays information about the allocated structures of a coupling facility and supports removal of this coupling facility from the sysplex. Usually, draining a coupling facility requires that at least one alternate coupling facility is enabled for the sysplex.

With INGCF DRAIN, you can perform the following sequence of tasks:

1. Rebuild all structures that can be rebuilt with user-managed or system-managed rebuild on an alternative coupling facility, and deallocate structure instances on the target CF that are being duplexed on another CF. For duplex structures, the duplexing process is stopped.

   The scope of the rebuild action depends partly on the release level of the systems from which the structures were allocated:

   - Structures that were allocated from a system with OS/390 2.7 or below can only be rebuilt if they have at least one active connector and all its active connectors support user-managed rebuild.
   - Structures that were allocated from a system with OS/390 2.8 or above can be rebuilt if they have an active connector and support either user-managed or system-managed rebuild, or if they have no active connector.

   Note: INGCF DRAIN rebuilds structures one at a time (SETXCF START,REBUILD,STRNAME=), not globally (SETXCF START,REBUILD,CFNAME=), and always on a CF that is different from the target CF (LOCATION=OTHER).

2. Force the deallocation of structures that have no active connectors and could not be rebuilt.
3. Disconnect the coupling facility from the systems to which it is connected.
4. Deactivate the coupling facility.

   Note: This task is unavailable when running on a z/OS image which runs under z/VM.

INGCF DRAIN ensures that the supported actions are carried out in the right order. Thus, for example, INGCF DRAIN lets you disconnect the coupling facility from the systems only after all structures of the coupling facility have been moved to another CF or have been deallocated. After each step, INGCF DRAIN presents the results of that step. You can then choose whether you want to initiate the next step.

For further information about the INGCF DRAIN command refer to *System Automation for OS/390 Operator's Commands*.

INGCF ENABLE

INGCF ENABLE is the counterpart of INGCF DRAIN. It supports integration of a new CF into a sysplex and reintegration of an existing CF into the sysplex, for example, after maintenance of the CF.

Note: INGCF ENABLE assumes that the receiver paths from the CF to the systems in the sysplex have been defined and activated. This requires a POR of the CPC on which the CF resides.

With INGCF ENABLE, you can perform the following sequence of tasks:

1. Activate the coupling facility.

   Note: This task is unavailable when running on a z/OS image which runs under z/VM.
2. Connect the systems of the sysplex with the coupling facility (sender paths).
3. Switch to another CFRM policy if
   - the target CF is not defined in the active policy, and
   - a policy is available that contains the target CF and definitions for all active CFs and all allocated structures.
4. Populate the target CF, that is, rebuild all those structures on the target CF, if this CF is the first usable one in the preference list, provided that this is not excluded by other requirements.

When the structures have been allocated on the target CF, INGCF ENABLE displays the result.

As with INGCF DRAIN, INGCF ENABLE ensures that the supported actions are carried out in the right order. Thus, you can only start populating the target CF after it has been connected to the systems in the sysplex.

For further information about the INGCF ENABLE command refer to the online help.

INGCF PATH

INGCF PATH lets you set the sender paths ONLINE or OFFLINE. The last sender path can only be set offline when no more structures are allocated. For further information about the INGCF PATH command refer to *System Automation for OS/390 Operator's Commands*.

INGCF STRUCTURE

INGCF STRUCTURE displays all the allocated structures of a coupling facility and information about their actual conditions. For a selected structure, you can:

- Display detail information
- Initiate a rebuild on another CF, depending on the rebuild pending status (PENDING calls LOCATION=NORMAL, otherwise LOCATION=OTHER)
- Force the deletion of the structure
- Start and stop duplexing

Rebuild and deletion can only be performed for structures with certain conditions.

For further information about the INGCF STRUCTURE command refer to the online help.

Customization

None. For information on how to control access to INGCF, refer to Appendix A of *System Automation for OS/390 Planning and Installation*.

Note that the ENABLE function requires that the active IODF is catalogued. otherwise, sender path information cannot be retrieved in certain situations.

Recovery Actions

Resolving a System Log Failure

SYSLOG message automation has been enhanced with a recovery function. Both functions (recovery and automation of message IEE043I) exist in parallel. Recovery takes place if the system log becomes inactive. It responds to message IEE037D following one of the messages IEE043I, IEE533E, or IEE769E, and it responds to message IEE041I. For details refer to "Enabling System Log Failure Recovery" on page 67. Except for the decision message, you can define individual action commands in the customization dialogs for the above messages.

Because the recovery and the former automation of message IEE043I affect the same resource SYSLOG, only one threshold can be defined in the policy SYSLOG THRESHOLDS. To allow the separate control of SYSLOG recovery from the former SYSLOG message automation, the new minor resource flag LOG has been introduced. For the run time environment, two thresholds are generated from the single threshold definition. The names of these thresholds correspond to the names of the minor resource flags.

Note:

Action commands that are executed for the old SYSLOG message automation are defined in the customization dialog using the entry SYSLOG in the messages policy for the entry type MVS Components. Action commands that are executed for the new SYSLOG recovery of message IEE043I are defined in the customization dialog in entry IEE043I in the same policy. If SYSLOG message and recovery commands are defined, both action commands will be issued, if message IEE043I followed by message IEE037D is trapped.

Customization: Automation of system log recovery is enabled through the SA OS/390 customization dialogs. For more details, see "Enabling System Log Failure Recovery" on page 67.

Resolving WTO(R) Buffer Shortages

When all WTO(R) buffers are in use, it is possible that commands can no longer be processed. To resolve this, there are several options: you can extend the buffer, change the properties of the affected consoles, or cancel jobs that issue WTO(R)s.

SA OS/390 provides recovery of buffer shortage in two stages. It first tries to extend the buffer and modify the console characteristics, if applicable. If this does not help, it then cancels jobs that issue WTO(R)s. You must specify which jobs can be canceled by SA OS/390 if there is a buffer shortage.

Customization: Automation of buffer shortage recovery is enabled using the SA OS/390 customization dialogs. For more information, see "Enabling WTO(R) Buffer Shortage Recovery" on page 69.

Handling Long-Running Enqueues (ENQs)

This recovery function lets you:

- Check which resources are blocked
- Customize automation to cancel or keep the jobs that block the resource
- Customize automation to dump the jobs before they are canceled

You can determine which resources you want to monitor. You can define a value for the maximum time a job can lock a resource while other jobs are waiting for it. If this amount of time is exceeded, recovery takes place. Identification of and elimination of these potential bottlenecks helps to reduce the risk of a Parallel Sysplex outage.

While the time definition describes an inclusion list, you also have the possibility to define an *exclusion list* of resources that are not monitored at all.

For more information about enabling the ENQ function, see "Enabling Long Running Enqueues (ENQs)" on page 74.

This function has now been extended to include automatic recovery of the SYSIEFSD family of resources.

SYSIEFSD Resource Recovery: The purpose of this function is to detect critical ENQ resources that, if held for extended periods of time, can cause commands to hang. Hung commands often result in multisystem outages. The focus of this function is on the SYSIEFSD family of resources that are involved in 98% of hung command outages:

- SYSIEFSD Q10 – this resource is required for every command. It is used to serialize changes to the CSCB chain. If any task gets this resource and then hangs, *all commands* will be locked out of the system. This also means that *all consoles* will be locked out of the system. This is because, as soon as a console issues a command after Q10 has hung, it will be waiting behind Q10, and that locks out the task that handles all MCS consoles. EMCS consoles will then also get locked out one by one as they issue a command and also get hung behind Q10. Actions taken to free up this hang cannot include issuing a command (for example, D GRS)—the task has to be terminated via CALLRTM.
- SYSIEFSD Q4 – this resource is used to serialize changes to the UCB by allocation and VARY command processing. Allocation obtains the resource as SHARED, while the VARY command obtains it exclusively. If a VARY command

hangs while holding this resource, all allocations will also hang. The VARY command that is hung can be displayed and abended with the CMDS command.

- SYSIEFSD VARYDEV – this resource is used in the processing of VARY commands. If the resource is hung, then all VARY commands will hang behind it. The VARY command that is hung can be displayed and abended with the CMDS command.
- SYSIEFSD CHNGDEVS – this resource is used in the processing of UNLOAD commands. If this resource is hung, then other UNLOAD commands will be hung behind it, and allocations may also hang. The UNLOAD command that is hung can be displayed and abended with the CMDS command.

If any of these resources do not execute within 10 seconds, they are considered to have hung.

Hung Command Recovery: The purpose of this function is to detect hung commands that often result in multisystem outages. Any commands shown by the MVS command CMDS SHOW that do not execute within 30 seconds are considered to have hung. Exceptions to this are the following commands:

- TRACE and DUMP are given a time limit of 5 minutes because they are interactive commands
- SET SLIP and SET MPF are excluded from hung command recovery
- A dump is made for all other commands that are abended because the reason for abending them is not known.

Customization: Automation of handling long-running enqueues is enabled through the SA OS/390 customization dialogs. For more details, see "Enabling Long Running Enqueues (ENQs)" on page 74.

Neither SYSIEFSD resource recovery nor hung command recovery need further customization.

System Removal

The purpose of this function is to isolate failed systems from a Parallel Sysplex by removing them as quickly as possible. It also ensures fast mean time to recovery (MTTR) for those system images that you wish to restart immediately if an unavoidable outage occurs.

Note: This function is unavailable when running on a z/OS image which runs under z/VM, even if the function is enabled.

In particular, the function automates the messages IXC102A and IXC402D.

The automation of the first message completes the Sysplex Failure Management (SFM). Under certain circumstances SFM cannot complete the isolation of a failed system. This is because SFM's HW isolation, resetting the channel subsystem (CSS) of the failed system, is driven through the CF. When connectivity between the system image and the coupling facility is lost, SFM cannot perform the hardware isolation (ISOLATE command) and defers resetting the system image until manual operator intervention occurs. Message IXC102A tells the operator to manually reset the HW and then reply "DOWN" to the message, after which SFM safely partitions the system image out of the sysplex. The longer the delay lasts, the more the components and applications that rely on XCF messaging are impacted. The delay can eventually lead to a sysplex outage when the failed system has I/O operations pending. Automation of this message minimizes the delay.

The second message has the same impact as the first one. However, this message indicates a possible temporary inoperative status of the system due to a missing status update. For this reason the automation gives the system the chance to recover before the removal takes place by replying ″INTERVAL=sss″ to the first occurrence of message IXC402D. The interval time is calculated as twice the SPINTIME value (defined in parmlib member EXSPAT*xx*) plus 5 seconds.

The automation does the removal of a system in two stages. The first stage clears any pending I/O operations by sending a hardware command to the Support Element. This requires information about the software running on the hardware. Because the system issuing message IXC102A or IXC402D does not necessarily have access to the hardware of the failed system, the automation needs predefined mapping between software and hardware. Depending on this mapping, it then routes the hardware command to the system that has access to the hardware of the failed system. For information about how to do the mapping refer to “Enabling System Removal” on page 70. For further information about the hardware requirements refer to *System Automation for OS/390 Planning and Installation*.

The second stage replies to the outstanding WTOR with ″DOWN″ triggering the removal of the system from the sysplex.

Customization: Automation of message IXC102A is enabled through the SA OS/390 customization dialogs. For more details, see “Step 3: Automating Messages IXC102A and IXC402D” on page 72.

Recovering Auxiliary Storage Shortage

With the automation of local page data sets, SA OS/390 prevents auxiliary storage shortage outages by dynamically allocating spare local page data sets when needed. The function checks which jobs cause the shortage condition and whether additional page data sets can be added. If this is not possible, the job that is causing the shortage will be canceled if this has been defined.

To enable local page data set automation customize the PAGTOTL parameter (defined in one of the IEASYS*xx* PARMLIB members used during IPL). Make sure to set the PAGTOTL parameter to a value greater than the number of local page data sets currently used.

Local page data sets must be defined in the master catalog and should not be SMS-managed. It is recommended to use preallocated local data sets instead of dynamically allocated ones. This makes the process faster because formatting newly allocated page data sets is timeconsuming (10sec./35MB). Each predefined local page data set should be allocated with 10% space of local page space currently used by the system. If predefined page data sets can no longer be allocated, new local page data sets will be created dynamically.

Customization: Automation of the recovery of auxiliary storage shortage is enabled through the SA OS/390 customization dialogs. For more details, see “Enabling System Removal” on page 70.

The IBM Health Checker for z/OS and Sysplex

The IBM Health Checker for z/OS and Sysplex is a tool that checks the current, active operating system (z/OS or OS/390) and sysplex settings and definitions for an image, and compares their values to those either suggested by IBM or defined by you, as your criteria. The objective of the HealthChecker is to identify potential problems before they impact your availability, or in worst cases, cause outages. The function produces reports (snapshots of your system) to help you analyze the

values defined for this system. SA OS/390 can automate the running of the checks **sysplex-wide** and provides an easy-to-use interface for viewing the report data.

General Prerequisites

The following operating systems are supported by the HealthChecker:

- OS/390 R10 or later
- all z/OS releases

The following system configurations are supported:

- XCFLOCAL is NOT supported
- MONOPLEX
- MULTISYSTEM

HealthChecker Best Practice Values

The values used by the HealthChecker are also referred to as best practices and originate from a variety of sources, including books and Web sites. However, the fact that the information comes from various sources can make it more difficult for you to ensure that your configurations reflect all of the suggestions. Using the HealthChecker means that this work is done for you. Another problem is keeping up with the changes that may have been made on your systems and ensuring that they still reflect either IBM's suggestions or your own criteria. To address this, you can ensure that the HealthChecker be run on demand or automatically, and hence easily determine if new values have introduced potential exceptions. We also realize that there are customer-unique and system-unique cases where the IBM suggestions are not appropriate. Therefore, you can either specify overrides to IBM values or suppress the running of a check. See "Customizing the IBM Health Checker for z/OS and Sysplex" on page 78 for details.

The HealthChecker checks the current values that are being used by your system; it does not check PARMLIB values. The scope of the checks are the local system where the function is run. It does not check values on other systems within the sysplex, although some values checked are sysplex-wide in scope. We recommend that you run the HealthChecker on all systems in your sysplex. In this case, all the systems in your sysplex will run the LOCAL checks (system-wide scope) but only one system in your sysplex will run the GLOBAL checks (sysplex-wide scope) in addition to the LOCAL checks. The way this latter system is determined is such that the HealthChecker function does an exclusive ENQ on a global GRS resource. The system that gets that LOCK will also do the Global checks.

When the HealthChecker function is enabled, it performs regular checks at predefined time intervals. The time intervals are defined individually for each check as part of IBM's best practices, although you can also override them. The checks are done based on IBM's best practices or your overrides. The HealthChecker implements the best practices in these ways:

1. Consolidates best practice values from multiple IBM sources
2. Reports on your configuration's active settings compared to IBM's suggestions, simplifying administration and operations
3. Reports on your configuration's active settings specific to any customer-specified preferences that can be used to override IBM values
4. Provides a mechanism for IBM to distribute updates to best practice values or to provide additional checks in a manner that is easily integrated into your environment

The basis for the values used by the IBM Health Checker for z/OS and Sysplex include:

- Parallel Sysplex and z/OS publications:
  - *z/OS MVS Setting Up a Sysplex*
  - *z/OS MVS Planning: Operations*, SA22-7601
  - *z/OS MVS Initialization and Tuning Reference*, SA22-7592
- Parallel Sysplex Availability Checklist

  The Parallel Sysplex Availability Checklist can be found at:

  `http://www.ibm.com/servers/eserver/zseries/pso/`
- ITSO Redbooks
  - *OS/390 Parallel Sysplex Configuration, Volume 1: Overview*, SG24-5637
  - *OS/390 Parallel Sysplex Configuration, Volume 2: Cookbook*, SG24-5638
  - *OS/390 Parallel Sysplex Configuration, Volume 3: Connectivity*, SG24-5639

  The Redbooks can be found at:

  `http://www.redbooks.ibm.com/`
- *z/OS Parallel Sysplex Test Report*, SA22-7663

  The Parallel Sysplex Test Report can be found at:

  `http://www.ibm.com/servers/eserver/zseries/zos/integtst/`
- Washington System Center Flashes

  Washington System Center Flashes can be found at:

  `http://www.ibm.com/support/techdocs/`

  Of particular interest for migration to a 64-bit environment is whitepaper WP100269, *z/OS Performance: Managing Processor Storage in a 64-bit environment*, and Washington System Center Flash 10086, *Software Capacity Planning: Migration to 64 bit Mode*.

INGPLEX BESTpractices

This command allows you to view the currently active best practices

INGHC

This command has two purposes:

1. Display the results of the checks
2. Trigger actions in the HealthChecker

Types of Reports: The HealthChecker reports reflect values at a point in time (snapshot). The report is comprised of a series of records in the System Logger. These records are comprised of the following components:

- Message text and explanation
- Actions that can be taken in case of an exception to address the exception
- IBM suggestions
- Reasons for IBM's suggestions

Types of Actions: The following type of action is available:

- You can request selected or all checks to be done immediately instead of waiting for the time interval of the respective checks to elapse. This is useful if you have made some change to your system and you want to immediately have these changes checked against the best practices. You can select individual systems in your sysplex or all at once.

Customization

Automation of the HealthChecker is enabled using the SA OS/390 customization dialogs. For more details, see "Customizing the IBM Health Checker for z/OS and Sysplex" on page 78.

You can override IBM's suggestions in order to:

- Specify your own values for a check
- Disable the running of a check

In order to activate these changes, use the INGAMS REFRESH command.

Hardware Validation

This function performs cross-validation of the hardware configuration mapped out in the customization dialogs against the actual hardware configuration that is running. This information is critical to accurately control logical partitions (LPARs) on any supported CPC within the HMC/SE LAN over the BCP Internal Interface.

Hardware validation uses the CPC name, Partition name and Partition number to ensure that the LPARs defined in the customization dialogs are on the correct CPC and located on the correct partition number. However, this helps only for coupling facilities because their partition identifiers must be defined in the active CFRM policy.

For MVS images, information from the HMC/SE (such as system name and sysplex name that are stored during initialization) is used to verify the corresponding customization dialog definitions. During initialization of the automation's Hardware Command Interface and just before a disruptive request is sent to a partition, new checks are made to ensure that everything matches correctly.

Note: Only active images can be verified. For inactive images we must still rely on definitions made in the customization dialogs.

An active system in this context is a system belonging to the same sysplex as the system that runs the hardware validation, that is SA OS/390 checks only systems and coupling facilities within its own sysplex.

Hardware validation runs on an SA OS/390 system primarily during startup, and subsequently when changes to the definition in the customization dialogs are applied through the ACF command (ACF COLD, and ACF REFRESH when any CPC or image data has changed). The validation checks the definitions of all registered systems, that is whenever an SA OS/390 system performs the hardware validation, it validates all systems and coupling facilities that are active in the sysplex at this point in time. Registered systems are systems running msys for Operations or SA OS/390 that have joined the same XCF group.

The validation of active systems and coupling facilities requires that the CPCs that host the active systems must all be defined in the customization dialogs.

The data for inactive systems cannot be verified. However, these definitions are checked for consistency across all registered systems. As soon as one of these inactive systems or coupling facilities joins the sysplex or is made available for use, the validation is run for the particular image only.

Retrieving actual hardware information can take up to 5 minutes per CPC depending on the model and its LPARs. During the time that the hardware

validation takes place all other hardware-related automation is either delayed or cannot be performed, depending on the type of recovery. For this reason the validation carries out ″delta″ processing. That is validating only the data that has changed. This also includes the absence of data resulting in terminating CPC connections when CPC definitions are missing that have been applied by a prior validation. The actions resulting from the validation are performed on ALL registered systems. This has two advantages:

- you don't need to recycle NetView for changes in hardware definitions.
- you only need to make the changes available to one system.

The first part of the hardware validation triggered by the ACF command or the automation startup determines what CPC connections must be terminated and initiated, namely in this sequence. The resulting actions are performed on all registered systems. When this step has been completed successfully the image validation is performed.

The image validation collects actual hardware information, and verifies the current hardware definitions against the actual data and the definitions found on all other registered systems. It informs you if:

- a real system or coupling facility could not be validated because either actual hardware information or user definitions are not available
- the image definitions could not be evaluated because the actual hardware information is not available
- the real system or coupling facility is not active and the image definitions of some of the registered systems are different
- any definition value has been corrected that was improperly defined or not defined at all

Changes in hardware definitions can be made available to all registered systems by simply invoking the command ACF COLD|REFRESH at only one of the these systems. There is one exception: the change of the authorization token value used for the communication with a particular CPC. A change of this value requires 3 steps:

1. In the first step you must remove the particular CPC definition and then invoke the ACF command as above.
2. When the command completes successfully the next step is to change the authorization token value of the CPC at the Support Element.
3. The final step is to define the CPC again with the new token value and invoke the ACF command again.

Note: This behavior of the ACF command applies to the hardware definitions ONLY.

The second part of the validation is triggered by either the message IXC517I that is issued when a coupling facility is made available for use, or by the automation itself when notified that a system joined the sysplex. Both trigger the automation to perform only the validation of the new system or coupling facility. Multiple occurrences of messages for the same system or coupling facility are ignored while this system or coupling facility is validated. In case of a new system, the advantage here is that the real hardware is validated before the system starts NetView and the automation. If this automation then detects no difference between its current definitions and the definitions of the other registered systems—which is the normal case—only a consistency check takes place. This check does not require any real hardware information.

Prerequisites

Hardware validation has the following prerequisites:

- SA OS/390 must have been initialized. For this reason the validation is delayed until the initialization has completed.
- All coupling facilities that are used in the sysplex must reside on a CMOS-S/390 G5 processor or higher. Only these processors return the partition identifier that is required for validating coupling facilities.
- The BCP Internal Interface must have been initialized to accept requests. Or, when unavailable, at least one other registered system must have access to the hardware. Registered systems are systems running msys for Operations or SA OS/390 that have joined the same XCF group.

Note: Hardware validation is not supported on MVS systems running under z/VM.

Miscellaneous

Recording IPL Information

With the INGPLEX IPL command you can record, view and compare the IPL information of the operating system. If a system does not behave after the IPL as expected, the IPL recording function enables you to identify parameters that were changed, for example, since the last IPL. The recording function enables you to compare different IPL scenarios. INGPLEX IPL is a tool that helps to identify the cause of problems. For further information about the INGPLEX IPL command refer to the online help.

System Dump Options

The enhanced INGPLEX functions allow you to control dump options sysplex-wide, for *registered* systems, that is, those on which the automation runs. For further information refer to the online help.

Multisystem Dumps

One of the enhanced INGPLEX functions provides an easy-to-use interface for multisystem dumps. For further information refer to the online help.

SLIP Traps

The enhanced INGPLEX functions also let you view, enable, disable, and delete SLIP traps defined in the sysplex. For further information refer to the online help.

Enabling Parallel Sysplex Automation (Hardware Automation)

To enable the enhanced Parallel Sysplex automation that SA OS/390 provides for recovery actions and coupling facility management (INGPLEX, INGCF, INGCFL, and INGHC commands), the following definitions must be made in the customization dialog.

Step 1: Defining the Processor

Use the customization dialog to define a processor of Entry Type PRO, as shown in Figure 21 on page 65. For more information, refer to the online help or the section "Creating a New Processor" in *System Automation for OS/390 Defining Automation Policy*.

```
  COMMANDS  HELP
-------------------------------------------------------------------------------
                              Define New Entry
Command ===> ____________________________________________________________

To define a new entry, specify the following information:
  Type                 : Processor
  Name . . . . . . . . . ________

  Processor Type . . . . . _______  (eServer 9672 9674 2003 3000)
  Mode . . . . . . . . . . ____     (LPAR ESA)

  Short Description  . . . ____________________________________________
  Extended Description . . ____________________________________________
                         . . ____________________________________________
                         . . ____________________________________________
                         . . ____________________________________________
                         . . ____________________________________________
```

Figure 18. Define New Entry Panel for Processors

Step 2: Using the Policy Item PROCESSOR INFO

Use the Processor Information panel, as shown in Figure 19, to define a processor using entry type PRO with connection type protocol INTERNAL. For more information, refer to the online help or the section "More about Policy Item PROCESSOR INFO" in *System Automation for OS/390 Defining Automation Policy*.

```
  COMMANDS  HELP
-------------------------------------------------------------------------------
                            Processor Information
Command ===> ____________________________________________________________

Entry Type     : Processor            PolicyDB Name   : DATABASE_NAME
Entry Name     : P9674LP              Enterprise Name : YOUR_ENTERPRISE

  Processor Type  . . . . . . . . _______   (eServer 9672 9674 2003 3000)
  Mode  . . . . . . . . . . . . . ____      (LPAR ESA)
  Connection Protocol . . . . . . ________  (INTERNAL NVC SNMP)
  CPC Name / NAU  . . . . . . . . ________
  Network Name  . . . . . . . . . ________
  Community Name / Auth Token . . ________
  Activation Reset Profile .  . . ________________
  Path Poll Frequency . . . . . . __          (0 to 99 minutes)
  Path Poll Retries . . . . . . . __          (0 to 99)

For SNMP connection, at least one address must be specified:
  TCP/IP Address or Hostname for Support Element
_______________________________________________________________________________
  TCP/IP Address or Hostname for HMC
```

Figure 19. Processor Information Panel

Step 3: Defining Logical Partitions

If the processor you have defined runs in LPAR mode, define its logical partitions using the LPAR Definitions panel, as shown in Figure 20 on page 65. For more information, refer to the online help or the section "More about Policy Item LPARS AND SYSTEMS" in *System Automation for OS/390 Defining Automation Policy*.

```
  COMMANDS  ACTIONS  HELP
------------------------------------------------------------------------------
                               LPAR Definitions            Row 1 to 13 of 20
Command ===>                                                 SCROLL===> PAGE

Entry Type : Processor             PolicyDB Name   : DATABASE_NAME
Entry Name : P9674LP               Enterprise Name : YOUR_ENTERPRISE

  Action     LPAR Name  Target Mode  Target System Name
  ________   ________   ________     ____________________
  ________   ________   ________     ____________________
  ________   ________   ________     ____________________
  ________   ________   ________     ____________________
  ________   ________   ________     ____________________
  ________   ________   ________     ____________________
  ________   ________   ________     ____________________
  ________   ________   ________     ____________________
  ________   ________   ________     ____________________
  ________   ________   ________     ____________________
  ________   ________   ________     ____________________
  ________   ________   ________     ____________________
  ________   ________   ________     ____________________
```

Figure 20. LPAR Definitions Panel

Step 4: Defining the System

Define a system using entry type SYS, and the Define New Entry panel, as shown in Figure 21. For more information, refer to the online help or the section "Creating a New System" in *System Automation for OS/390 Defining Automation Policy*.

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                               Define New Entry
Command ===> ____________________________________________________________________

To define a new entry, specify the following information:
   Type. . . . . . . . . . . System
   Name. . . . . . . . . . . ____________________
                                                                  More:     +
   Operating system. . . . . MVS      MVS VM VSE LINUX CF
   MVS SYSNAME . . . . . . . ________ MVS system name (MVS systems only)
   Image/ProcOps name. . . . ________ Image/ProcOps name

Specify information for NMC Focal Point Communication (MVS systems only):
   Heartbeat Interval. . . . 5        1 - 60   (minutes)
   Missing Heartbeat Delay . 30       1 - 3600 (seconds)

The following fields are for reference:
   Short Description . . . . ____________________________________________
   Extended Description. . . ____________________________________________________
                           . . ____________________________________________________
                           . . ____________________________________________________
```

Figure 21. Define New Entry Panel for Sysplex Groups

Step 5: Connecting the System to the Processor

Connect this system to the processor that you defined in "Step 2: Using the Policy Item PROCESSOR INFO" on page 64 and to its logical partition (if you set the processor mode as LPAR).

Connect this system to the sysplex or standard group.

Note: MVS SYSNAME and the Image/ProcOps Name *must* be the same.

Miscellaneous

Restriction::

The MVS SYSNAME must not begin with a number. The MVS SYSNAME may normally begin with a number, but in this case, because it must be the same as the Image/ProcOps Name, which cannot begin with a number, then this restriction also applies to the MVS SYSNAME.

Enabling Continuous Availability of Couple Data Sets

Couple data sets (CDSs) contain important information about how to manage certain aspects of your sysplex. For example, the SFM CDS (sysplex failure management couple data set) defines how the system manages system and signalling connectivity failures and PR/SM (processor resource/system manager) reconfiguration actions.

The following couple data sets are particularly important for the functioning of your Parallel Sysplex:

- The SYSPLEX couple data set, which defines the systems and the XCF groups of the sysplex
- The CFRM couple data set, which defines the coupling facilities and structures of the sysplex

It is recommended that you define alternate couple data sets for all couple data sets in your sysplex. These alternate couple data sets serve as backups when the primary CDS fails.

With the customization dialog you can specify a series of spare volumes for every CDS type, for example, SYSPLEX, ARM, CFRM. The first volume in the series is used to create an alternative CDS if one of the primary alternate CDSs fails.

In the customization dialog you define the potential alternate couple data sets using the *Group* entry type. Select a sysplex group, then select its policy item SYSPLEX (define sysplex policy) from the panel ***Policy Selection***.

Defining the Policy Item SYSPLEX

A panel as shown in Figure 22 on page 67 is displayed if you select policy item SYSPLEX from the ***Policy Selection*** panel for sysplex groups.

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                          Sysplex Policy Definition
Command ===> ____________________________________________________________

Entry Type : Group                PolicyDB Name   : DATABASE_NAME
Entry Name : SYSPLEX_GROUP_01     Enterprise Name : YOUR_ENTERPRISE
                                                          More:     +
Sysplex Name. . . . . . . . . . . ________
Sysplex Timer Monitoring. . . . . . YES       YES NO
Number Monitored Sysplex Timers . . 2         1   2
Temporary Data Set HLQ. . . . . . . ____________________________________
                                    Data set HLQ (max. 17 chars)
Started Task Job Name . . . . . . . ________
Couple Data Set HLQ . . . . . . . . ________  ________  ________

CDS type    Alternate volumes                 Desired monitoring
                                              (PRIMARY ALTERNATE NONE)
Sysplex     ______ ______ ______ ______       PRIMARY
            ______ ______ ______ ______
ARM         ______ ______ ______ ______       PRIMARY
            ______ ______ ______ ______
```

Figure 22. Sysplex Policy Definition Panel for Sysplex Groups

For a description of this panel refer to the online help or the section "More About Policy Item SYSPLEX" in *System Automation for OS/390 Defining Automation Policy*.

Enabling System Log Failure Recovery

The SA OS/390 customization dialog supports the automation of the system log failure recovery by defining commands for the following messages:

- IEE041I
- IEE043I
- IEE533E
- IEE769E

Use the *MVS Component* entry type to specify the commands that will be issued in case of a SYSLOG problem. Select the MESSAGES/USER DATA policy item of a selected *MVS Component* policy object to display the *Message Processing* panel. Enter`CMD` in the ***Action*** column and the message ID in the ***Message ID*** column, as shown in Figure 23 on page 68.

Miscellaneous

```
  COMMANDS  ACTIONS  HELP
------------------------------------------------------------------------------
                               Message Processing
Command ===> ________________________________________________  SCROLL===> PAGE

Entry Type  : MVS Component        PolicyDB Name   : DATABASE_NAME
Entry Name  : MVS_COMPONENT        Enterprise Name : YOUR_ENTERPRISE

Subsystem   : MVSESA

Enter messages issued by this resource that will result in automated actions.
Actions:  CMD = Command  REP = Reply  CODE = CODE  USER = User defined values

Action   Message ID                                Cmd  Rep  Code  User
         Description
CMD      IEE041I
```

Figure 23. Message Processing Panel

Press Enter to display the *CMD Processing* panel, as shown in Figure 24. On this panel you specify the MVS command that will be executed in case of message IEE041I. For example, enter `MVS VARY SYSLOG,HARDCPY` to have the SYSLOG receive the hardcopy log. (This action is recommended by IBM.)

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                                 CMD Processing             Row 1 to 2 of 20
Command ===> ________________________________________________  SCROLL===> PAGE

Entry Type : MVS Component        PolicyDB Name   : DATABASE_NAME
Entry Name : MVS_COMPONENT        Enterprise Name : YOUR_ENTERPRISE

Subsystem  : MVSESA
Message ID : IEE041I

Enter commands to be executed when resource issues the selected message.

Pass/Selection Automated Function/'*'
Command Text

MVS VARY SYSLOG,HARDCPY
```

Figure 24. CMD Processing Panel

In case of message IEE043I, the IBM recommended action is to enter the MVS command `MVS WRITELOG START` to restart the system log.

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                                CMD Processing              Row 1 to 2 of 20
Command ===> ____________________________________________  SCROLL===> PAGE

Entry Type : MVS Component         PolicyDB Name   : DATABASE_NAME
Entry Name : MVS_COMPONENT         Enterprise Name : YOUR_ENTERPRISE

Subsystem  : MVSESA
Message ID : IEE043I

Enter commands to be executed when resource issues the selected message.

Pass/Selection Automated Function/'*'
Command Text

_____ _______
MVS WRITELOG START
_____________________________________________________________________________
_____________________________________________________________________________

_________     _______
_____________________________________________________________________________
_____________________________________________________________________________
```

Figure 25. CMD Processing Panel

For the remaining messages repeat the steps as shown in the preceding panels.

You can use the customization dialog ***Minor Resource Selection*** to disable the system log recovery by setting the automation flag of the minor resource LOG to NO. For details refer to the section "More About Policy Item MINOR RESOURCE FLAGS" in *System Automation for OS/390 Defining Automation Policy*.

Enabling WTO(R) Buffer Shortage Recovery

The SA OS/390 customization dialog supports the automation of WTO(R) buffer shortage recovery.

> **Note:**
> SA OS/390 2.1 provides automation for WTO buffer recovery in policy item WTOBUF RECOVERY of the *MVS Component* entry type. However, use this policy only to define the WTO buffer shortage recovery process for SA OS/390 2.1 without. For SA OS/390 2.2, use policy item MESSAGES/USER DATA to obtain the SA OS/390 facility to cancel jobs.

When using the *MVS Component* entry type (MVC), you can specify jobs that will be canceled or kept in case a WTO(R) buffer shortage is threatening. The jobs that you select for cancellation will then no longer issue WTO(R)s.

Select the MESSAGES/USER DATA policy item of a selected *MVS Component* policy object to display the Message Processing panel.

EnterCODE in the ***Action*** column and WTOBUF in the ***Message ID*** column as shown in Figure 26 on page 70.

```
  COMMANDS  ACTIONS  HELP
------------------------------------------------------------------------------
                              Message Processing
Command ===> _____________________________________________  SCROLL===> PAGE

Entry Type  : MVS Component         PolicyDB Name   : DATABASE_NAME
Entry Name  : MVS_COMPONENT         Enterprise Name : YOUR_ENTERPRISE

Subsystem   : MVSESA

Enter messages issued by this resource that will result in automated actions.
Actions:  CMD = Command  REP = Reply  CODE = CODE  USER = User defined values

Action   Message ID                                  Cmd  Rep  Code  User
         Description
CODE     WTOBUF
         ____________________________________________
_______  ___________________________________
         ____________________________________________
_______  ___________________________________
         ____________________________________________
_______  ___________________________________
```

Figure 26. Message Processing Panel

After pressing Enter, the Code Processing panel is displayed, as shown in Figure 27.

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                               Code Processing              Row 1 to 6 of 20
Command ===> _____________________________________________  SCROLL===> PAGE

Entry Type : MVS Component         PolicyDB Name   : DATABASE_NAME
Entry Name : MVS_COMPONENT         Enterprise Name : YOUR_ENTERPRISE

Subsystem  : MVSESA
Message ID : WTOBUF

Enter the value to be passed to the calling CLIST when this resource
issues the selected message and the following codes are contained in
the message.

Code 1          Code 2          Code 3           Value Returned
CIC*            *               _____________    KEEP
IMS*            *               _____________    KEEP
*               *               _____________    CANCEL
_____________   _____________   _____________    ____________________________
_____________   _____________   _____________    ____________________________
_____________   _____________   _____________    ____________________________
```

Figure 27. Code Processing Panel

For more information about this panel, refer to the online help or to the section "More About Policy Item MESSAGES/USER DATA" in *System Automation for OS/390 Defining Automation Policy*.

Enabling System Removal

The SA OS/390 Parallel Sysplex enhancements help you to resolve pending I/Os for systems being removed from the sysplex.

Because the automation must know where the system is located to send the command to the appropriate Support Element, you must use the customization dialog to define its hardware configuration.

Step 1: Defining the Processor and System

The processor and system must be defined as described in "Enabling Parallel Sysplex Automation (Hardware Automation)" on page 63.

Step 2: Defining the Application with Application Type IMAGE

Use entry type APL to define a new application with Application Type IMAGE and subsystem name that is the same as the Image Name of the system that this application represents (as defined in "Step 4: Defining the System" on page 65).

Use entry type APL and select policy item APPLICATION INFO for your system. On the panel *Application Information* you can define a new application type IMAGE. For more information, refer to the online help or the section "Policy Items for Applications" in *System Automation for OS/390 Defining Automation Policy*.

```
     COMMANDS  HELP
-------------------------------------------------------------------------------
AOFGLN00                        Define New Entry
Command ===> ____________________________________________________________

To define a new entry, specify the following information:
   Type. . . . . . . . . . Application
   Application Name. . . . ____________________
   Subsystem Name. . . . . ___________
                                                              More:     +
   Object Type . . . . . . INSTANCE   CLASS     INSTANCE
   Application Type . . .  IMAGE      STANDARD IMAGE JES2 JES3
                                      CICS IMS DB2 OPC USS
                                      (Only the value STANDARD can be changed)
                                      (once, all others cannot be changed)
                                      (after the application has been created)
   Subtype . . . . . . . . ___________ (Only for CICS, IMS, OPC or DB2)
   Job Name. . . . . . . . ____________
   Scheduling Subsystem. . ________    MSTR, JES Subsystem or blank
   JCL Procedure Name. . . ____________________________
   MVS Automatic Restart
     Management Element
     Name. . . . . . . . . ____________________________
                                      (Only if the application uses)
                                      (MVS Automatic Restart Management)
   WLM Resource Name . . . ____________________________________

   Short Description . . . ___________________________________________
   Extended Description. . ____________________________________________________
                         . . ____________________________________________________
                         . . ____________________________________________________
                         . . ____________________________________________________
```

Figure 28. Definition of Application Type IMAGE

Because the application has been defined as type IMAGE, the jobname is set by default to the subsystem name and cannot be changed.

The Subtype, Scheduling Subsystem, JCL Procedure Name, ARM Element Name, and WLM Resource Name are forced to be blank.

Some other definitions in the policy item AUTOMATION INFO are also defaulted (see Figure 29 on page 72):

- the Job Type is defaulted to NONMVS
- the Monitor Routine is defaulted to INGMTSYS if nothing is specified
- the Eternal Startup is defaulted to ALWAYS if the Monitor Routine is INGMTSYS

- the Eternal Shutdown is defaulted to ALWAYS if the Monitor Routine is INGMTSYS

```
  COMMANDS  HELP
-------------------------------------------------------------------------------
                       Application Automation Definition
Command ===> ____________________________________________________________________

Entry Type : Application           PolicyDB Name   : DATABASE_NAME
Entry Name : KEY3IMAGE             Enterprise Name : YOUR_ENTERPRISE

Subsystem  : KEY3IMAGE
Description:
Job Name   : KEY3


Job Type . . . . . NONMVS      Job properties (MVS NONMVS TRANSIENT)
Transient Rerun  . ___         Transient Jobtype can be rerun (YES NO)
Command Prefix   . ____________________________
                               Enter console command prefix (above)
Message Prefix . . ____________________________________________________________
                               Enter one or more prefixes (above)
Sysname  . . . . . ________     System name used by the application

Start on IPL . . . ____        Start with Netview init (YES NO NONE blank)
Start on Recycle . ____        Start with Sys-Ops recycle (YES NO NONE blank)
Start Timeout  . . ________    Time allowed to reach "UP" status (hh:mm:ss)

Monitor Routine. . INGMTSYS    Routine used for monitoring (name NONE)
Periodic Interval. _____       Periodic monitoring interval (hh:mm NONE)

Restart Option . . _________   Restart Circumstances (ALWAYS ABENDONLY NEVER)
External Startup . ALWAYS      External Startup (INITIAL ALWAYS NEVER blank)
External Shutdown. ALWAYS      External Shutdown (FINAL ALWAYS NEVER blank)

Shut Delay . . . . ________    Time between attempts to shutdown (hh:mm:ss)
Term Delay . . . . ________    Time for termination cleanup (hh:mm:ss)
```

Figure 29. Application Automation Definition

For more information, refer to the online help or the section "More About Policy Item AUTOMATION INFO" in *System Automation for OS/390 Defining Automation Policy*.

Step 3: Automating Messages IXC102A and IXC402D

With SA OS/390 2.2, you can automate messages IXC102A and IXC402D to avoid sysplex outages.

Note: The following figures show examples for defining commands and codes for message IXC102A.

You can specify one of the following four hardware commands for each system in the sysplex that is automated.

- SYSRESET [CLEAR]
- DEACTIVATE
- ACTIVATE [P(image_profile_name)]
- LOAD [P(load_profile_name)] [CLEAR]

where

CLEAR indicates that the storage will be cleared

P specifies the profile to be used. The name can consist of up to 16 alphanumeric characters. If the parameter is omitted, the last profile is used.

Note:
The following restriction applies to the hardware commands ACTIVATE and LOAD:

Both commands invoke processor functions that can cause asynchronous events such as operator messages at BCP (Basic Control Program) Internal Interface initialization time or processor hardware wait states. Currently, the BCP Internal Interface does not allow the monitoring and control these events

Use policy item MESSAGES/USERDATA of the SA OS/390 customization dialog to define commands and codes for message IXC102A and IXC402D. Enter `CMD` in the **Action** column and `IXC102A` in the **Message ID Description** column (or `IXC402D` for IXC402D message automation), as shown in Figure 30. For more information, refer to the online help or the section "More About Policy Item MESSAGES/USER DATA" in *System Automation for OS/390 Defining Automation Policy*. The definitions here also apply to message IXC402D.

```
  COMMANDS  ACTIONS  HELP
------------------------------------------------------------------------------
                              Message Processing            Row 1 to 4 of 21
Command ===> ____________________________________________  SCROLL===> PAGE

Entry Type  : Application           PolicyDB Name   : DATABASE_NAME
Entry Name  : KEY3IMAGE             Enterprise Name : YOUR_ENTERPRISE

Subsystem   : KEY3IMAGE

Enter messages issued by this resource that will result in automated actions.
Actions:  CMD = Command  REP = Reply  CODE = CODE  USER = User defined values

Action   Message ID                                  Cmd  Rep  Code  User
         Description
CMD      IXC102A
         ____________________________________________
______   ____________________________________
         ____________________________________________
______   ____________________________________
         ____________________________________________
______   ____________________________________
```

Figure 30. Sample Panel for Defining IXC102A Automation

Pressing Enter will bring up the CMD Processing panel, as shown in Figure 31 on page 74. Use this panel to specify a valid command for the image and a "Pass/Selection" value that must match the "Value Returned" definition specified on the *Code Processing* panel.

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                                CMD Processing              Row 1 to 2 of 20
Command ===> ____________________________________________  SCROLL===> PAGE

Entry Type  : Application           PolicyDB Name   : DATABASE_NAME
Entry Name  : KEY3IMAGE             Enterprise Name : YOUR_ENTERPRISE

Subsystem   : KEY3IMAGE
Message ID : IXC102A

Enter commands to be executed when resource issues the selected message.

Pass/Selection Automated Function/'*'
Command Text
ACTCODE
 LOAD P(LOADPROF) CLEAR
_____________________________________________________________________________
_____________________________________________________________________________


_________     ________
_____________________________________________________________________________
_____________________________________________________________________________
```

Figure 31. Sample Panel for Command Processing

On the *Code Processing* panel, as shown in Figure 32, specify the following:

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                                Code Processing             Row 1 to 6 of 20
Command ===> ____________________________________________  SCROLL===> PAGE

Entry Type  : Application           PolicyDB Name   : DATABASE_NAME
Entry Name  : KEY3IMAGE             Enterprise Name : YOUR_ENTERPRISE

Subsystem   : KEY3IMAGE
Message ID  : IXC102A

Enter the value to be passed to the calling CLIST when this resource
issues the selected message and the following codes are contained in
the message.

Code 1          Code 2          Code 3          Value Returned
IXC102A         BCPII           _______________ ACTCODE
_______________ _______________ _______________ _____________________________
_______________ _______________ _______________ _____________________________
_______________ _______________ _______________ _____________________________
_______________ _______________ _______________ _____________________________
_______________ _______________ _______________ _____________________________
```

Figure 32. Sample Panel for Code Processing

If you want to automate messages IXC102A and IXC402D using the Parallel Sysplex enhancements, you must enter `IXC102A` for Code 1 and `BCPII` for Code 2. Refer to “Important Processor Operations Considerations” on page 81 for more information.

Enabling Long Running Enqueues (ENQs)

If you automate long running ENQs, you must define the following:

- The resource(s) being checked
- The time frame when a long ENQ is detected

In addition, the following definitions can be made:

- The names of jobs that should be cancelled or kept when detecting a long ENQ

- The title of the dump taken before the job is cancelled
- The default storage areas to be dumped
- Symbol definitions to be used when the dump specifications are provided by a PARMLIB member

Use the dialog support via entry type GRP to define the following policies:

- Resource definition
- JOB/ASID definitions
- IEADMC*xx* symbols

Step 1: Defining Resources

A panel as shown in Figure 33 is displayed if you select policy item RESOURCE DEFINITIONS from the Long Running Enqueue Policy section of the ***Policy Selection*** panel for sysplex groups. For more information, refer to the online help or the section "More About Policy Item RESOURCE DEFINITIONS" in *System Automation for OS/390 Defining Automation Policy*.

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                     Long Running ENQ Resource Definition    Row 1 to 9 of 20
Command ===> _____________________________________________  SCROLL===> PAGE

Entry Type : Group                 PolicyDB Name   : DATABASE_NAME
Entry Name : SYSPLEX_GROUP_01      Enterprise Name : YOUR_ENTERPRISE

Define the major and minor resource names together with their wait time
(30-999 sec) after that an ENQ is treated as a long running ENQ.

Major       Minor                                                Waittime/EXCL
Resource/*  Resource/*                                          (30-999 s)
________    ________________________________________________     ____
________    ________________________________________________     ____
________    ________________________________________________     ____
________    ________________________________________________     ____
________    ________________________________________________     ____
________    ________________________________________________     ____
________    ________________________________________________     ____
________    ________________________________________________     ____
________    ________________________________________________     ____
```

Figure 33. Long Running ENQ Resource Definition Panel for Sysplex Groups

Step 2: Making Job/ASID Definitions

A panel as shown in Figure 34 on page 76 is displayed if you select policy item JOB/ASID DEFINITIONS from the Long Running Enqueue Policy section of the ***Policy Selection*** panel for sysplex groups. For more information, refer to the online help or the section "More About Policy Item JOB/ASID DEFINITIONS" in *System Automation for OS/390 Defining Automation Policy*.

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                     Long Running ENQ Job/ASID Definitions   Row 1 to 3 of 20
Command ===> _____________________________________________  SCROLL===> PAGE

Entry Type : Group                PolicyDB Name   : DATABASE_NAME
Entry Name : SYSPLEX_GROUP_01     Enterprise Name : YOUR_ENTERPRISE

Dump Options . . . _____________________________________________________________
Dump Title . . . . _____________________________________________________________
_______________________________________________________________________________

Define the jobs and address space IDs which can or cannot be cancelled with or
without dump (KEEP/NODUMP/DUMP) or their dump IDs in case the job/ASID is one
of those responsible for the long running ENQ.
A default specification with Job Name/ASID=* must exist.

Job Name/   Handling
ASID/*      (KEEP/NODUMP/DUMP/dumpids)

________    ____________________________________________________________________
________    ____________________________________________________________________
________    ____________________________________________________________________
```

Figure 34. Long Running ENQ Job/ASID Definitions Panel for Sysplex Groups

Step 3: Defining IEADMC*xx* Symbols

A panel as shown in Figure 35 is displayed if you select policy item IEADMCxx SYMBOLS from the Long Running Enqueue Policy section of the ***Policy Selection*** panel for sysplex groups. For more information, refer to the online help or the section "More About Policy Item IEADMCxx SYMBOLS" in *System Automation for OS/390 Defining Automation Policy*.

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                       Long Running ENQ IEADMCxx Symbols     Row 1 to 9 of 20
Command ===> ___________________________________________  SCROLL===> PAGE

Entry Type : Group                PolicyDB Name   : DATABASE_NAME
Entry Name : SYSPLEX_GROUP_01     Enterprise Name : YOUR_ENTERPRISE

Define the IEADMCxx symbols together with their substitution value for
selective jobs or address space IDs.

Job Name/*  Symbol and Value
ASID/*      (&symbol.='value')

________    ____________________________________________________________________
________    ____________________________________________________________________
________    ____________________________________________________________________
________    ____________________________________________________________________
________    ____________________________________________________________________
________    ____________________________________________________________________
________    ____________________________________________________________________
________    ____________________________________________________________________
________    ____________________________________________________________________
```

Figure 35. Long Running ENQ IEADMCxx Symbols Panel for Sysplex Groups

Enabling Auxiliary Storage Shortage Recovery

To prevent auxiliary storage shortage outages you can predefine local page data sets, using the SA OS/390 customization dialog for entry type GRP to define the following:

- local page data set

- job definitions

Step 1: Defining the Local Page Data Set

A panel as shown in Figure 36 is displayed if you select policy item LOCAL PAGE DATA SET from the Local Page Data Set Policy section of the ***Policy Selection*** panel for sysplex groups. For more information, refer to the online help or the section "More About Policy Item LOCAL PAGE DATA SET" in *System Automation for OS/390 Defining Automation Policy*.

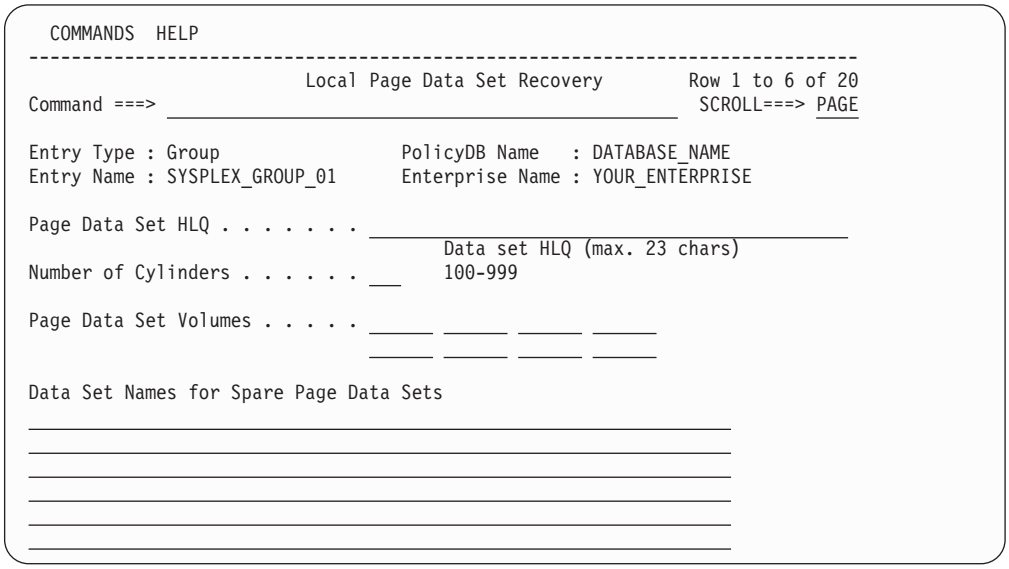


Figure 36. Local Page Data Set Recovery Panel for Sysplex Groups

Step 2: Defining the Handling of Jobs

A panel as shown in Figure 37 on page 78 is displayed if you select policy item JOB DEFINITIONS from the Local Page Data Set Policy section of the ***Policy Selection*** panel for sysplex groups. For more information, refer to the online help or the section "More About Policy Item JOB DEFINITIONS" in *System Automation for OS/390 Defining Automation Policy*.

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                  Local Page Data Set Recovery Job Definition Row 1 to 8 of 20
Command ===> _____________________________________________  SCROLL===> PAGE

Entry Type : Group                PolicyDB Name   : DATABASE_NAME
Entry Name : SYSPLEX_GROUP_01     Enterprise Name : YOUR_ENTERPRISE

Define the jobs which can or cannot be cancelled (KEEP/CANCEL) in case the job
is one of those jobs responsible for the shortage condition.
A default specification with Job Name=* must exist.

Job Name/*  Handling
            (KEEP/CANCEL)

________    ______
________    ______
________    ______
________    ______
________    ______
________    ______
________    ______
________    ______
```

Figure 37. Local Page Data Set Recovery Job Definition Panel for Sysplex Groups

Customizing the IBM Health Checker for z/OS and Sysplex

There is no dedicated customization dialog support for automation of the HealthChecker, however, you can specify your HealthChecker user overrides as UET data. To do this:

1. Specify a new entry on the panel *UET entry type selection* with UET Entry `HEALTHCHECK` and UET Type `USERPRACTICES`.
2. Define UET keyword data for each User override (see Figure 38 on page 79 and Figure 39 on page 80 for examples) so that each entry is in the form L*xx*, where *xx* is a number that can be viewed as a (logical) line number, that is:
   - Two adjacent entries need not have sequential numbers
   - There must not be any gaps in these numbers for the entries as a whole, that is, if there are *n* entries then the range for all *xx* must be 1 to *n*
   - These line numbers define the sequence in which the override statements are processed. This sequence must be in accordance with the HealthChecker's syntax requirements.
   - If you want to insert a new line in the UET entries, you do not have to reenter all of the existing entries. Consider an example where you want to change the checking interval for the SYSCONS_MSCOPE check (L60 in Figure 38 on page 79) to 1 hour. To do this:
     a. change keys L61–L69 to L62–L70, that is, increment the logical line numbers by 1 (so that L61 becomes L62, and so on)
     b. then add a new keyword entry, `L61 TIMEINT(01:00)`, as the last entry (remember that consecutive keyword entries do not have to be sequential)

```
                        UET Keyword-Data Specification        Row 16 from 447
Command ===>                                                  SCROLL===> CSR

Entry Type : User E-T Pairs        PolicyDB Name   : MSYSRESV_21
Entry Name : HC_BACKUP             Enterprise Name : MSYSRESV_21
UET Entry  : HEALTHCHECK           UET Type        : USERPRACTICES

Action   Keyword/Data(partial)
         L69
         "CHECK(SYSCONS_MASTER)"
         L68
         "REASON('SYSCONS SHOULD ... D012103');"
         L67
         "DATE(20030121)"
         L66
         "CHECK(SYSCONS_PD_MODE)"
         L65
         "REASON('IF SYSCONS .... D012103');"
         L64
         "DATE(20030121)"
         L63
         "CHECK(SYSCONS_ROUTCODES)"
         L62
         "REASON('IF SYSCONS .... - D012103');"
         L61
         "DATE(20030121)"
         L60
         "CHECK(SYSCONS_MSCOPE)"
         L59
         "REASON('EMCS CONSOLES WITH HARDCOPY ... - D012103');"
         L58
         "DATE(20030121)"
         L57
         "CHECK(EMCS_HARDCOPY)"
         L56
         "REASON('ROUTCODE(ALL) AND NON-LOCAL ... - D012103');"
         L55
         "DATE(20030121)"
         L54
         "CHECK(EMCS_MSCOPE_AND_ROUTCODES)"
         L53
         "REASON('NEEDED IN EMERG. - D012103');"
         L52
         "DATE(20030121)"
         L51
         "CHECK(SYSPLEX_MASTER)"
         L50
         "REASON('NOT REALY - D012103');"
         L49
         "DATE(20030121)"
         L48
         "CHECK(CONSOLE_ROUTCODE_11)"
         L47
         "REASON('AVOIDS LONG CHAINS D012003');"
         L46
         "DATE(20030120)"
         L45
         "CHECK(AMRF_AND_MPF_CONSISTENT)"
         L44
         "REASON('AVOIDS OVERLOADING ... D012003');"
         L43
         "DATE(20030120)"
         L42
         "CHECK(CONSOLE_MSCOPE_AND_ROUTCODES)"
         L41
         "REASON('NEEDED FOR DCCF D012003');"
         L40
         "DATE(20030120)"
```

Figure 38. UET Keyword-Data Specification Example

```
                              Edit UET Keyword-Data
Command ===>

To change keyword-data pair, specify the following:

Keyword
Data
L69
"CHECK(SYSCONS_MASTER)"
```

Figure 39. UET Keyword-Data Entry for L69

Defining Common Automation Items

Two new definitions are introduced relating to utilities running as a started task. The first one (Temporary Data Set HLQ/TEMPHLQ) replaces the usage of the first qualifier of the status file. The second definition (Started Task Job Name/STCJOBNM) allows the unique assignment of started task job names scheduled by the automation in case you have dedicated job name assignments that conflict with the procedure names provided by the automation.

It is recommended that you define the Temporary Data Set HLQ/TEMPHLQ. If it is not defined, the automation uses the first qualifier of the AOF status file.

You can define both of these items using the Sysplex Policy Definition panel, as shown in Figure 40, that is displayed if you select the policy item SYSPLEX from the ***Policy Selection*** panel for sysplexes. For more information, refer to the online help or the section "More About Policy Item SYSPLEX" in *System Automation for OS/390 Defining Automation Policy*.

```
  COMMANDS  HELP
------------------------------------------------------------------------------
                           Sysplex Policy Definition
Command ===> ____________________________________________________________

Entry Type : Group                PolicyDB Name   : DATABASE_NAME
Entry Name : SYSPLEX_GROUP_01     Enterprise Name : YOUR_ENTERPRISE
                                                              More:     +
Sysplex Name. . . . . . . . . . . _______
Sysplex Timer Monitoring. . . . . . YES       YES NO
Number Monitored Sysplex Timers . . 2         1   2
Temporary Data Set HLQ. . . . . . . _____________________________________
                                    Data set HLQ (max. 17 chars)
Started Task Job Name . . . . . . . _______
Couple Data Set HLQ . . . . . . . . _______  _______  _______

CDS type    Alternate volumes                 Desired monitoring
                                              (PRIMARY ALTERNATE NONE)
Sysplex     ______ ______ ______ ______       PRIMARY
            ______ ______ ______ ______
ARM         ______ ______ ______ ______       PRIMARY
            ______ ______ ______ ______
CFRM        ______ ______ ______ ______       PRIMARY
            ______ ______ ______ ______
LOGR        ______ ______ ______ ______       PRIMARY
            ______ ______ ______ ______
SFM         ______ ______ ______ ______       PRIMARY
            ______ ______ ______ ______
WLM         ______ ______ ______ ______       PRIMARY
            ______ ______ ______ ______
```

Figure 40. Sysplex Policy Definition Panel for Sysplex Groups

Important Processor Operations Considerations

Currently, the IXC102A automation and Coupling Facility activation or deactivation is the product automation that uses the new BCP (Basic Control Program) Internal Interface to control the processor hardware.

If you use the automation capabilities of SA OS/390 processor operations in your environment, make sure they do not conflict with the automation supplied by the Parallel Sysplex enhancements.

This book explains how to automate processor operations controlled resources. The section "Message Automation for IXC102A" on page 118 describes the automation of message IXC102A using the processor operations facilities to perform processor functions such as ACTIVATE or SYSTEM RESET.

With the Parallel Sysplex enhancements, IXC102A and IXC402D automation uses the BCP Internal Interface, which is currently not compatible with processor operations.

If you want to use the IXC102A automation that is supplied as part of the Parallel Sysplex enhancements, make sure there is no processor operations related IXC102A automation defined in your automation policy.

Likewise, if you want to continue to use the processor operations based automation of messages IXC102A and IXC402D, the IXC102A automation flag provided by the Parallel Sysplex enhancements must be disabled.

Processor operations, which is a Focal Point type function, allows you to monitor and control processor hardware, including Coupling Facility images, from a single NetView, the processor operations Focal Point.

The BCP Internal Interface of the Parallel Sysplex enhancements allows you to perform hardware operations from each NetView in your sysplex member, as long as its processor hardware supports this. Refer to *System Automation for OS/390 Planning and Installation* for more information.

Important Processor Operations Considerations

Chapter 6. How to Add a Message to Automation

Use the NetView message automation table and the SA OS/390 command set to implement console automation. You can automate the routine functions an operator performs when a particular message is generated.

The process to add a new message to automation consists of two steps:

1. In the customization dialog you specify how SA OS/390 should respond to a message issued by an application (for example, respond with command processing or with reply processing). This step is described in *System Automation for OS/390 Defining Automation Policy*.
2. In the NetView message automation table, you enter the message and the routine SA OS/390 should use to process the user reaction as specified in the customization dialog. How to achieve this, is the topic of this chapter.

Note:

If you use product automation (DB2, CICS, IMS, OPC automation), take care that message entries for these products do not conflict with other entries in your NetView message automation table.

If the message is issued by an application previously undefined to SA OS/390, you must add the application to automation. See Chapter 3, "How to Add a New Application to Automation", on page 21.

Using Multiple NetView Message Automation Tables

SA OS/390 has the possibility to load its shipped NetView message automation table that you do not have to modify (except the synonym member AOFMSGSY) and you can load another message table in parallel (see also "Multiple Master Automation Tables" on page 13.

Thus the SA OS/390 message table fragments can be easily serviced without affecting your own NetView message automation table. Additionally you do not have to care about messages that are used by SA OS/390 and you do not have to specify CONTINUE(Y) for such messages.

You need to specify the name of your NetView message automation table in the customization dialog. This NetView message automation table will be loaded at SA OS/390 initialization unless another NetView message automation table is specified via the AOF603D reply.

If no message automation table is specified in the customization dialog, SA OS/390 will load the INGMSG01 NetView message automation table.

Step 1: Automating the Response to the Message

You can automate the response to a message issued by an application or MVS component using the appropriate policy items in entry types *Application* and *MVS Component* in the customization dialog. Choose the type of response you want to make to the message. SA OS/390 allows you to:

- Issue a command in response to the message, "Responding with a Command"
- Issue a reply in response to the message, "Responding with a Reply"
- Match codes in the message with codes defined in the automation policy, and, if a match occurs, pass an action keyword back to the calling automation procedure, "Responding with Code Matching"
- Specify user data that will be stored in the ACF and can be retrieved from there by common routines to be used in your own automation procedures, "Creating User Specific Message Processing" on page 85.

For all four methods of message processing, you start with the ***Message Processing*** panel that you can invoke from both entry types as described in *System Automation for OS/390 Defining Automation Policy*.

Responding with a Command

You can have SA OS/390 issue a command as a reaction to a message issued by an application. In the customization dialog you specify your command that you want to issue. Use the ISSUECMD generic routine from the NetView message automation table for that purpose. See also *System Automation for OS/390 Programmer's Reference* for information on the options that ISSUECMD provides for issuing multiple pass commands.

Responding with a Reply

You can have SA OS/390 respond with a *reply* as a reaction to a message issued by an application. In the customization dialog you specify your reply that you want to issue. Use the ISSUEREP generic routine from the NetView message automation table for that purpose. See also *System Automation for OS/390 Programmer's Reference* for information on ISSUEREP.

Responding with Code Matching

The steps to define code matching in response to a message are:

1. Add the code matching information to the automation policy using the customization dialog.
2. Either use the generic routine ISSUEREP for replies (see also *System Automation for OS/390 Programmer's Reference* for information about generic routines or create an automation procedure that uses the value specified with the codes in the automation policy.

Expanded descriptions of these steps are in the following sections.

Creating an Automation Procedure to Use Code Matching Information

If the message response you are automating performs code matching for a message, create an automation procedure that uses the code values and actions defined in the code matching information in the automation policy. This automation procedure must contain a call to the CDEMATCH common routine to handle the code matching function. For more information on this common routine, see *System Automation for OS/390 Programmer's Reference*.

Details on creating automation procedures are in Chapter 8, "How to Create Automation Procedures", on page 95. More specific details on using the code matching feature in automation procedures are in *System Automation for OS/390 Programmer's Reference* and *System Automation for OS/390 Defining Automation Policy*.

Note: For elementary code matching, ISSUEREP may be called with CODE= parameters specified. ISSUEREP then calls CDEMATCH and issue any match as a command or reply respectively.

Creating User Specific Message Processing

With the user-specific message processing method, you can create your own processing information in the automation control file. SA OS/390 just stores the data that you specify on the appropriate panel in the customization dialog into the automation control file. However, SA OS/390 does not interpret or check your input. You can use common routines, for example, ACFFQRY or CDEMATCH, to work with that data. These routines retrieve your input from the automation control file for your use. See *System Automation for OS/390 Programmer's Reference* for information on available common routines.

Step 2: Adding an Entry in the NetView Message Automation Table

To activate the message response handling process you must add an entry in the NetView message automation table that calls the appropriate SA OS/390 generic routine, common routine or automation procedure when the message occurs.

When creating NetView automation tables, order the NetView automation table entries with the most frequently matched statements first because of performance. Take care that further entries for the same message must be reachable. You may also want to keep a backup copy of the automation table as it existed before installing SA OS/390.

What you specify in the NetView message automation table depends on the type of message response, see the following subsections:

- "Messages Issued by an Application or MVS Component"
- "Messages Issued by a Processor Operations Target System" on page 86

Messages Issued by an Application or MVS Component

If a message response is a command, add a call to the ISSUECMD generic routine.

An example NetView message automation table entry for ISSUECMD is:

```
IF MSGID = 'IST020I' & DOMAINID = %AOFDOM%
THEN EXEC(CMD('ISSUECMD AUTOTYP=START')  ROUTE(ONE *)) CONTINUE(Y);
```

For more information, see *System Automation for OS/390 Programmer's Reference*.

Note: This entry means that the SA OS/390 response to this message is controlled by the Start automation flag setting for VTAM® minor resource VTAM.IST020I.

If the message response is a reply, add a call to the ISSUEREP generic routine.

An example NetView message automation table entry for ISSUEREP is:

```
IF MSGID = 'AHL125A' & DOMAINID = %AOFDOM%
THEN EXEC(CMD('ISSUEREP AUTOTYP=START')  ROUTE(ONE %AOFOPWTORS%));
```

For more information, see *System Automation for OS/390 Programmer's Reference*.

If the message response is code matching, add a call to the automation procedure that uses the code values and actions specified in the code matching information of

the automation policy. This automation procedure must contain a call to the CDEMATCH common routine to handle the code matching.

For more information on CDEMATCH, see *System Automation for OS/390 Programmer's Reference*.

For example, the following NetView message automation table entry calls an automation procedure named MYPROC when message ABC123 occurs:

```
IF MSGID = 'ABC123' & DOMAINID = %AOFDOM%
THEN EXEC(CMD('MYPROC')  ROUTE(ONE *));
```

The automation procedure should retrieve the message and parse it to extract the codes that will be searched on. You can do this either in the automation table or in the CLIST. For example, to code a CDEMATCH for the message `ABC123I JOB myjob SENT DATASET name TO user`, you could either:

1. Parse in the automation procedure.
   - Automation Table entry.

     ```
     IF MSGID = 'ABC123I' & DOMAINID = %AOFDOM%
     THEN
        EXEC(CMD('MYPROC') ROUTE(ONE *));
     ```

   - Automation procedure code

     ```
     'GETMLINE LINE 1'
     Parse Var line . . jobname . . dataset . user .


     CDEMATCH ENTRY=MVSESA,TYPE=ABC123I,CODE1='||jobname||,
             ',CODE2='||dataset||',CODE3='||user
     ```

2. Parse in the automation table:
   - Automation table entry

     ```
     IF MSGID='ABC123I' & TOKEN(3) = SVJOB & DOMAINID = %AOFDOM%
       &TOKEN(6) = SVDSN & TOKEN(8) = SVUSER
     THEN
       EXEC(CMD('MYPROC ' SVJOB ' ' SVDSN ' ' SVUSER)
           ROUTE(ONE *));
     ```

   - Automation procedure code

     ```
     Arg jobname dataset user .


     'CDEMATCH ENTRY=MVSESA,TYPE=ABC123I,CODE1='||jobname||,
              ',CODE2='||dataset||',CODE3='||user
     ```

Parsing in the automation table has the advantage that you are able to use the same automation procedure for several different messages if you pass the entry and the type in as parameters.

Messages Issued by a Processor Operations Target System

When a target system issues a message, the message is forwarded to the processor operations focal point system. The focal point system repackages the message within an SA OS/390 ISQ900I message, an ISQ901I message, or both, and routes the message to the appropriate task:

- ISQ900I messages are routed to SA OS/390 processor operations autotasks. If you want automation that you write to receive ISQ900I messages, use the ISQEXEC command to run the automation in a target control task. For information about using the ISQEXEC command, see section *Sending an Automation Routine to a Target Control Task* in "Issuing Other OCF Commands" on page 100

page 100. Your NetView message automation table entries for SA OS/390 should acknowledge the ISQ900I identifier for all target system messages forwarded to the processor operations focal point system. You can specify your ISQ900I automation table entries to be target system specific, however, this is not recommended.

- ISQ901I messages are routed to all logged-on operators identified as interested operators by the ISQXMON command or marked as such in the customization dialog.

For information about the ISQEXEC and ISQXMON commands, see *System Automation for OS/390 Operator's Commands*.

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

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

For example, if a NetView connection forwards the message `IEA101A SPECIFY SYSTEM PARAMETERS` from the operating system to the focal point system, SA OS/390 creates one or both of the following SA OS/390 messages:

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

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

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

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

Sample NetView Automation Table Entries

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

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

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

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

Message ISQ211I

Some SA OS/390 commands attempt to lock and unlock ports. Where an operator owns the lock for a port, the SA OS/390 unlock command, ISQXUNL, returns RC=12 associated with message ISQ211I `Unable to unlock` *`target name console`*.

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

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

Processor Operations Command Messages

Some SA OS/390 commands run on the target system. The message returned from these commands indicates only that the support element was told to schedule the operation. Consequently, the operation at the target system may not complete even though the SA OS/390 message indicates a successful completion.

SA OS/390 acknowledges only that the command was successfully forwarded to the support element. An unsuccessful operation at the target system generates an unsolicited message that the support element forwards to the focal point system in an ISQ900I message. Schedule your automation from the message that positively indicates that a target system operation did or did not complete.

The SINGSAMP SA OS/390 sample library contains the PL/I source code for several automation routines that issue responses to selected messages. You can select the response that is most appropriate for your enterprise. You can also use them as models to create your own automation routines. The following list summarizes these routines, the messages they respond to, and the responses they issue initially:

SINGSAMP Member	Routine	Description
INGEI120	ISQI120	Responds to the following messages: **IEA120A** `Device ddd volid read, reply cont or wait.` **IOS120A** `Device ddd shared (PR volid not read.)` `the recovery task, reply cont or wait.` Issues the following response to the target: `CONT`
INGEI357	ISQI357	Responds to the following message: **IEE357A** `Reply with SMF values or U.` Issues the following response to the target: `U`
INGEI426	ISQI426	Responds to the following message: **$HASP426** `Specify options - subsystem_id.` Issues the following response to the target: `WARM,NOREQ.`
INGEI502	ISQI502	Responds to the following message: **ICH502A** `Specify name for primary/backup` `RACF data set sequence nnn or none.` Issues the following response to the target: `NONE`

SINGSAMP Member	Routine	Description
INGEI877	ISQI877	Responds to the following message: **IEA877A** Specify full DASD SYS1.DUMP data sets to be emptied, tape units to be used as SYS1.DUMP data sets or GO. Issues the following response to the target: GO
INGEI956	ISQI956	Responds to the following message: **IEE956A** Reply - ftime = hh.mm.ss, name = operator,reason = (ipl,reason) or u. Issues the following response to the target: U

The SA OS/390 automation table entries in the ISQMSG0 member of the SINGNPRM data set include inactive entries that call these automation routines. To incorporate these routines into your automation, do the following:

1. Remove the comments from the corresponding automation table entries for the messages that initiate the automation routines you want to use. If you perform these steps as part of the initial SA OS/390 installation, make these changes before you incorporate the SA OS/390 entries. If you do this after the initial SA OS/390 installation, change the NetView message automation table.
2. Code the routines you will be using to issue the responses you want.
3. Compile the PL/I source code for the routines you want to use, and link the resulting object code to your PL/I library.
4. Recycle the NetView program to activate the new entries.

For automation processing to occur, each message in the NetView message automation table at the focal point system and at each target system must be made available to the system's NetView program. In OS/390, MPF controls message availability to the NetView program. Examine the MPF list member in the SYS1.PARMLIB data set to ensure that the necessary messages are marked for automation. For target systems using other operating systems, check the message suppression facilities used on those systems.

Testing Messages

SA OS/390 provides a collection of NetView automation table entries for your SA OS/390 configuration. NetView automation table entries are in the AOFCMD member of the SA OS/390 SINGNPRM installation data set. When these entries are moved to your NetView automation table, they may need additional editing.

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

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

```
IEA320A RESPECIFY PARAMETERS OR CANCEL
```

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

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

With SA OS/390 installed, the following message appears when forwarded from a PC

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

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

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

After the SA OS/390 entries are added, the NetView automation table includes the following entry:

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

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

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

When you use the second example, both commands are scheduled.

If your NetView message automation table tests the text of SA OS/390 messages, the message format must match the character case for which you test. This can be done by requiring all sites to use the same format for their messages, or by duplicating message table entries in uppercase and in mixed formats.

Step 3: Assigning the Message to the Automation Environment

Messages issued by an application are assigned automatically to an AOFWRKxx automated function, so you need not do the assignments on a message ID basis.

The steps to assign a new message to your automated operations environment are:

1. Code an entry for the message in the MPF list, if necessary.
2. Add or update an automated function definition using the customization dialog.
3. Add a NetView automation operator ID, if necessary.

Expanded descriptions of these steps are in the following sections.

Coding an Entry for the Message in the MPF List

If required, add a message processing facility (MPF) list entry specifying the message ID and the AUTO(YES) parameter value. If the default for the message is already AUTO(YES), bypass this step.

If you are automating a message, you probably also want to suppress the message from operator consoles. To mark a message for suppression, code SUP(YES) in the MPF list entry for the message. For more information on coding MPF list entries, see *z/OS MVS Initialization and Tuning Reference*.

Adding a NetView Automation Operator ID

If you create a *new* automated function definition, you must also add a NetView automation operator ID.

Add the operator ID or IDs specified on the ***Automation Operator Definition*** panel into NetView DSIPARM data set member DSIOPF. This member is the NetView operator identification member. In the operator definition statement specify profile and logon initial command list values to use for this operator ID. Other operator ID definitions already in NetView DSIPARM data set member DSIOPF may specify some profile and logon initial command list values, from which you can select appropriate values for this new operator ID.

Note: You will probably need to change an include (DSIOPFU), rather than the DSIOPF member itself.

Step 4: Building the New Automation Definitions

When you are finished using the customization dialog to add message response and automation operator information to the automation policy, you need to build the system operations control files. The complete description of how to build and distribute these files is provided in *System Automation for OS/390 Defining Automation Policy*.

The SA OS/390 build function will place the new automation definitions in the data set defined in the ***ACF and AMC Build Parameters*** panel.

Copy the new automation definitions into the SA OS/390 NetView DSIPARM concatenation in the NetView startup procedures, or concatenate it to the NetView DSIPARM data set.

Step 5: Loading the Changed Automation Environment

To reload the AMC file and the automation control file with your changes, now perform the actions as described in “Step 9: Reloading Tables” on page 27.

Chapter 7. How to Write Your own Monitor Routines

SA OS/390 determines the status of an application by running a routine identified by the policy administrator in the customization dialog. The routine can be specified for an individual application (refer to *System Automation for OS/390 Defining Automation Policy*), and a default monitor routine can be specified for all applications on an entire system (see policy item AUTOMATION INFO in the customization dialog). The routines that can be specified as application monitors are listed in the following. A description of their purposes is given in *System Automation for OS/390 Defining Automation Policy*.

- AOFADMON
- AOFAJMON
- AOFATMON
- AOFAPMON
- AOFCPSM
- AOFUXMON
- EVJRSMON
- ISQMTSYS

SA OS/390 expects certain return codes from any monitor routine, either from SA OS/390 provided ones or from your own routines. So this section presents the template for user-written monitor routines that return a local return code in the variable `lrc`. This template assumes that you invoke your routine with parameter `job` which is the name of the application that you want to monitor.

```
Arg job
:
:
check job status /* for example: MVS D A,job */
:
:
select
   when job active   then lrc = 0
   when job starting then lrc = 4
   when job inactive then lrc = 8        /* this rc is optional */

   otherwise lrc = 12                    /* error occurred      */
end

exit lrc
```

Chapter 8. How to Create Automation Procedures

You may need to perform additional customization to extend your system automation as new needs develop. For example, you may need to automate additional messages or add a subsystem or application to the automation.

The main tasks involved in extending automation include:
- Adding or changing values via the SA OS/390 customization dialog
- Building a new automation control file
- Adding or changing entries in the message processing facility (MPF) message list and the NetView message automation table.
- Adding new resources to the status display facility (SDF)
- Reloading the changed files and tables, such as the MPF list, NetView message automation table, and automation control file, to enable the new or changed automation

Programming Additional SA OS/390 Automation Procedures

You can write additional automation procedures to supplement the basic automation procedures supplied by SA OS/390. For example, you may want to develop procedures to automate an application used exclusively on your system or to perform specialized automated operations for a subsystem.

SA OS/390 generic routines and common routines provide routines to perform basic functions such as logging messages and checking automation flags. You can use them in your own automation procedures.

"How Automation Procedures Are Structured" on page 96 describes how to structure your automation procedures. Refer to *System Automation for OS/390 Programmer's Reference* for detailed descriptions and examples of the generic routines, common routines and file manager commands you can use in your automation procedures.

How Generic Routines and Common Routines Are Used in Automation Procedures

SA OS/390 generic routines and common routines are convenience routines that provide your automation procedures with a simple, standard way of interfacing with the automation control file, automation status file, and NetView log file. It is strongly recommended that you use these routines wherever possible in your own code.

For a list of provided generic routines and common routines and their detailed description refer to *System Automation for OS/390 Programmer's Reference*.

How Automation Procedures Are Called

There are several ways to call an automation procedure including:

- Calling the automation procedure from the NetView message automation table using SA OS/390 generic routines
- Keying in the automation procedure name or its synonym into a NetView command line

- Calling the automation procedure from another program
- Starting the automation procedure with a timer
- Starting the automation procedure with the NetView EXCMD command
- Starting the automation procedure on an automation operator with the SA OS/390 AOFEXCMD command routine
- In the customization dialog, entering your automation procedure name into the ***Command text*** or ***Command*** field of the following entry types:
  - ***Application***
  - ***MVS Component***
  - ***Timers***

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

How Automation Procedures Are Structured

Automation procedures can be written in NetView CLIST language or in REXX. The recommended structure of such automation procedures should contain three main parts as shown in the subsequent figures. These parts are:

1. perform initialization processing
2. determine whether automation is allowed
3. perform automation processing.

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

Figure 41 illustrates the structure of automation procedures for system operations and Figure 42 on page 97 illustrates the structure of automation procedures for processor operations.

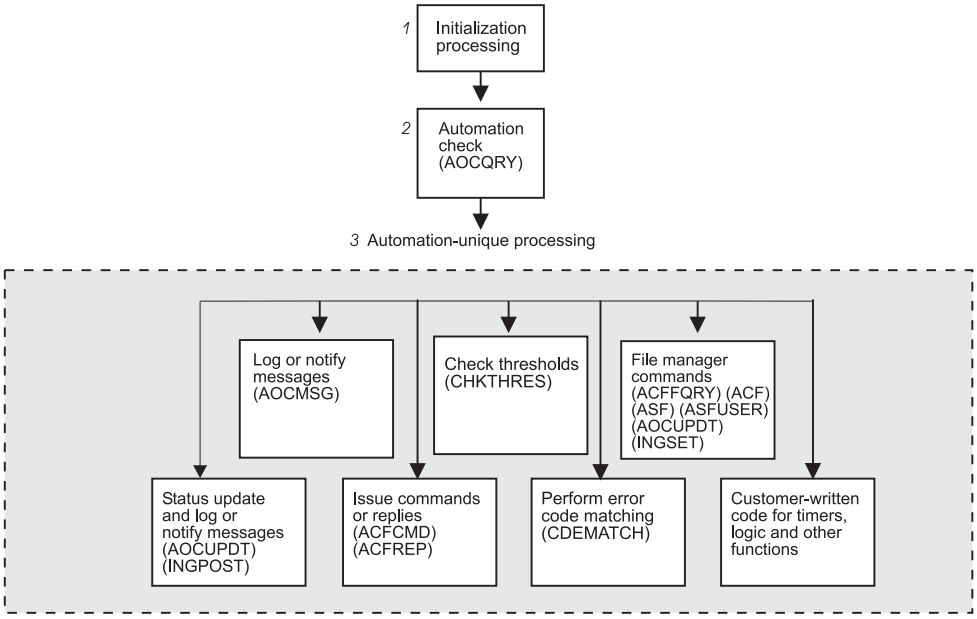


Figure 41. Automation Procedures for System Operations

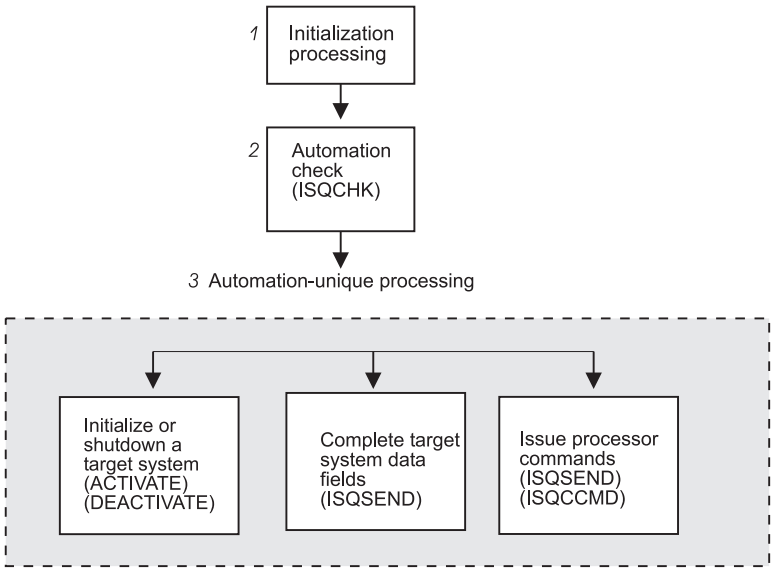


Figure 42. Automation Procedures for Processor Operations

Performing Initialization Processing

Initialization processing may not be required for simple automation procedures.

Initialization processing is responsible for:

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

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

  See Appendix A, "Table of External Global Variables", on page 141 and Appendix B, "Global Variables to Enable Advanced Automation", on page 149 for descriptions of global variables. In REXX, you must GET the first time you reference the globals, and PUT when you update them.
- Checking to see if debugging (general or CLIST-specific) is on.
- Issuing debugging messages, if debugging is turned on.
- Validating the automation procedure call.

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

For more information on coding automation procedure initialization sections, refer to "Example Automation Procedure" on page 104, to *Tivoli NetView for OS/390 Customization Guide* and to *Console Automation Using NetView: Implementing*.

Determining Whether Automation Is Allowed

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

Refer to *System Automation for OS/390 Programmer's Reference* for more information on coding the automation check routine.

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

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

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

Performing Automation Processing

Automation processing is performed by any combination of SA OS/390 routines and your own code. The following documentation gives more information on coding automation procedures. It is divided into two subsections:

- "Automation Processing in System Operations"
- "Automation Processing in Processor Operations" on page 100

Automation Processing in System Operations

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

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

For more information, see *System Automation for OS/390 Programmer's Reference*.

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

AOCMSG will:

- format a message for display or logging
- issue messages as SA OS/390 notification messages to notification operators.

For more information, see *System Automation for OS/390 Programmer's Reference*.

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

- Issue one or more commands in response to a message.
- Issue a single reply in response to a message.

- Use the step-by-step (PASS) concept to react to or recover from an automation event.

ACFCMD issues one or more commands. It supports both a single reaction and the step-by-step (PASS) concept. For more information, see *System Automation for OS/390 Programmer's Reference*.

ACFREP issues a single reply. It supports both a single reaction and the step-by-step (PASS) concept. For more information see *System Automation for OS/390 Programmer's Reference*.

In many cases you may be able to use the ISSUECMD and ISSUEREP generic routines which also support single and pass processing.

Checking Thresholds: You can check and update thresholds by calling the CHKTHRES common routine. Use CHKTHRES to track and maintain a threshold, and to change the recovery action based on the threshold level exceeded. For more information see *System Automation for OS/390 Programmer's Reference*.

Checking Error Codes: You can check error codes by calling the CDEMATCH common routine. Use CDEMATCH to compare error codes in a message to a set of automation-unique error codes to determine the action to take. For more information, see *System Automation for OS/390 Programmer's Reference*.

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

Using File Manager Commands: You can use file manager commands to access SA OS/390 control files such as the automation control file and automation status file. Use ACF if you need to load or display the automation control file. Use ACFFQRY to query the automation control file quickly. Use ASF to display the automation status file. Use ASFUSER to modify the automation status file fields reserved for your own information. For more information, see *System Automation for OS/390 Programmer's Reference*.

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

- Issuing commands and trapping responses.

  You can issue commands and trap responses using the NetView WAIT or PIPE commands. You may need to use these commands in your code if it is necessary to check the value or status of a system component or application before continuing processing. For more information, see *Tivoli NetView for OS/390 Customization Guide*
- Setting Common Global and Task Global values to control processing.

  You can set Common and Task Global values by using NetView commands. You may need to set these values if it is necessary to set a flag indicating progress, message counts, and other indicators that must be kept from one occurrence of a message to the next. See *System Automation for OS/390 Defining Automation Policy* for a table of all externalized SA OS/390 global variables.

  Also refer to the discussion of CGLOBALs and TGLOBALs in *Tivoli NetView for OS/390 Customization Guide*
- Setting timer delays to resume processing.

You can set timer delays by using the NetView AT, AFTER, and EVERY commands. You can use these commands when an automation procedure must either resume processing or initiate another automation procedure after a given time to do additional processing. For example, you could use these commands to perform active monitoring of subsystems. For more information, see the discussion of AT, AFTER, and EVERY commands in *Tivoli NetView for OS/390 Automated Operations Network User's Guide*

Automation Processing in Processor Operations

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

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

Shutting Down a Target System: If your routines need to shut down a target system, issue the ISQCCMD DEACTIVATE OCF command. Before issuing the command to close the target system, shut down all of your functioning subsystems. This avoids any unexpected situations at the target system.

Issuing Other OCF Commands: All OCF commands supported by processor operations can be issued from automation routines. See *System Automation for OS/390 Operator's Commands* for details about these commands.

Reserved SA OS/390 Commands: The SA OS/390 commands ISQISUP, ISQISTAT, ISQCMMT, ISQSTRT, ISQXIPM, ISQGPOLL, and ISQGSMSG are not intended for your use. Do not use these in your automation routines. Unexpected results may occur.

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

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

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

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

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

The NetView program has priority defaults established during its initialization. Usually, everything running under NetView has a low priority. You can use the NetView DEFAULTS command to see what the settings are, but you should not change them. For SA OS/390 command processing to be serialized as designed, all commands used in SA OS/390 must have a priority setting of "low". If you change the priorities or have more than one priority for commands used in SA OS/390, the difference in the priorities may defeat the serialization that results from the architecture of the target control task.

Sending an Automation Routine to a Target Control Task: If you run the same series of SA OS/390 commands regularly, you can program the commands into a NetView automation routine. Follow the guidelines you use for any NetView automation routine.

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

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

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

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

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

For more information about the ISQEXEC command, see *System Automation for OS/390 Operator's Commands*.

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

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

You can use the ISQXLOC and ISQXUNL commands within automation routines to ensure that they complete without interference from other tasks. For automation routines that issue a number of SA OS/390 commands, put the following command after the ISQEXEC command and near the beginning of the routine:

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

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

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

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

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

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

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

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

How to Make Your Automation Procedures Generic

By using the SA OS/390 common routines, you can make your own automation procedures generic. A generic automation procedure comprises three parts. For each part, there are special common routines that help you to fulfill your tasks:

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

Evaluation
What should be done? Use common routine CDEMATCH.

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

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

AOCQRY

- check if the resource is controlled by SA OS/390

- check if automation is allowed

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

...

CDEMATCH

- code matching (able search in ACF)

- find out required action

...

ACFCMD/ACFREP

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

Figure 43. Skeleton of an Automation Procedure

For more information on the mentioned common routines refer to *System Automation for OS/390 Programmer's Reference*. For more information on command processing or reply processing refer to *System Automation for OS/390 Defining Automation Policy*.

Processor Operations Commands

Whenever possible, your automation routines should make use of SA OS/390's processor operations OCF commands, also called common commands. These commands are independent of the hardware type of the target system's processor. Therefore, the use of these commands minimizes the need for changes to your automation routines if you need to add new processors to your configuration. See *System Automation for OS/390 Operator's Commands* for a detailed description of the processor operations commands.

Developing Messages for Your Automation Procedures

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

Some SA OS/390 facilities and commands you can use to develop messages include:

- The AOCMSG common routine (see *System Automation for OS/390 Programmer's Reference*).
- The AOCUPDT common routine (see *System Automation for OS/390 Programmer's Reference*).

The following steps summarize the message development process.

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

Put an entry for the message in a DSIMSG data set. This data set must be identified in a DSIMSG data definition (DD) name.

3. Use the AOCMSG common routine to issue the message (see *System Automation for OS/390 Programmer's Reference*).
4. Add an entry for the message to your production copy of the NetView DSIMSG data set.

Example AOCMSG Call

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

Entries for messages in DSIMSG member DSIABC12 are as follows:

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

Your automation procedure contains the following AOCMSG call:

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

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

```
ABC123I THE EAGLE HAS LANDED
```

Note that the message is defined with a 10 and a 40 between the message ID and the first word of the message. These are the SA OS/390 message classes to which the message belongs. When the message is issued a copy is sent to every notification operator who is assigned class 10 or class 40 messages.

Refer to *Tivoli NetView for OS/390 Customization Guide* for further information on developing new messages.

Adding NetView Message Automation Table Entries

When adding NetView message automation table entries it is important that you do not duplicate any entries or add entries that will conflict with existing entries. Use the AUTOTBL command to load the changed NetView message automation table. See "Step 5: Loading the Changed Automation Environment" on page 91 for more information.

Example Automation Procedure

This section provides an example of an application program that handles an OS/390 message. The automation procedure uses a subset of the SA OS/390 common routines or generic routines.

```
/* Example SA OS/390 Automation Procedure */

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

```
2   Parse source .. ident .

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

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

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

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

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

10      ehkvar1 = save_job
        ehkvar2 = save_asid
        "GLOBALV PUTT EHKVAR1 EHKVAR2"
11      cmd = 'ACFCMD ENTRY='||AOFSYSTEM||',MSGTYP='||save_msg
        cmd
        svretcode = rc
        If loc.0debug = 'Y' Then
          "PIPE LIT /Called ACFCMD; Return Code was "||svretcode||"/" ,
            "| LOGTO NETLOG"

        /* ------------------------------------------------------------ **
        ** Check return code from ACFCMD                                **
        ** 0 = ok                      1 = no commands found in ACF  **
        ** 4 = bad parms               5 = errors/timeout            **
        ** ------------------------------------------------------------ */
12      If svretcode > 1 Then Do
```

```
        "AOCMSG "loc.0me",206,,"time()",,,'"cmd"',RETCODE="svretcode
        lrc = 1
      End
    End
End /* End of Select svretcode */

13 Exit lrc

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

Notes on the Automation Procedure Example

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

2 This step defines the identity of the automation procedure.

3 This step handles the debug and trace settings (refer to "Using AOCTRACE to Trace Automation Procedure Processing" on page 110.

4 Save the NetView message variables the automation procedure uses.

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

6 This section performs the automation check:

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

7 Issue message AOF206I if call to AOCQRY fails.

8 Issue message AOF580I if automation flag is off.

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

10 Set EHKVARn variables for ACFCMD.

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

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

12 Issue message AOF206I if call to ACFCMD fails.

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

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

Installing Your Automation Procedures

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

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

For more information on preparing your code for use and installing it, refer to *Tivoli NetView for OS/390 Customization Guide*

Testing and Debugging Automation Procedures

This section describes SA OS/390 and NetView facilities you can use for testing automation procedures, including:

- SA OS/390 AOCTRACE operator facility
- NetView testing and debugging facilities
- SA OS/390 assist mode

The Assist Mode Facility

SA OS/390 provides the *assist mode* facility, so that you can verify actions of automation procedures and automation policy before letting them run in a completely automated environment.

When assist mode is on, actions normally taken by SA OS/390 automation procedures, such as issuing a command or reply or calling a common routine, are instead written to a log file or displayed through SDF. Operators using SDF can view assist mode displays to validate the actions. The operator can choose to have SA OS/390 do the scheduled automated action, change it, or end it. For example, SDF might indicate that the current automation procedure issues a command to restart TSO. You can then continue the automation action as it is, change the automation action, or cancel it.

Assist settings are associated with specific automation flags (Initstart, Start, Recovery, Shutdown or Restart). The assist setting used for any action is determined by the automation flag that is checked to see if the action is permitted.

You can activate assist mode using the automation flag panels of the customization dialog. You can turn assist mode on and off for all automated resources, resource groups, or individual resources. You can also use the SETASST command dialog to change the assist mode settings for a resource.

Cases where you might want to use assist mode include:

- During early stages of developing and using your automation policy
- After changing your automation policy, such as after adding an application to automation
- After adding a new automation procedure to the SA OS/390 code

Note: Assist mode is unavailable for subsystems controlled by IMS Automation.

Using Assist Mode to Test Automation Procedures

Assist mode can help detect problems with your automation procedures before they are added to your production code. Assist mode works by intercepting commands and replies before they are issued through NetView. The intercepted commands and replies, as coded in the automation control file, are reformatted into message AOF317A, sent to the NetView log and, optionally, to the SDF. Message AOF317A has the following format:

```
AOF317A time : ASSIST: KEY: {type|keyword} - COMMAND: text
```

Message AOF317A contains detailed information regarding:

- The time the message was generated
- The type field from the automation control file entry that defines the command
- The command selection field from the automation control file entry
- The actual command or reply

Assist mode can be enabled and disabled using the SETASST command. The DISPASST command can be used to view the current assist settings. You can also define permanent assist settings in your policy, but it is recommended that you use SETASST.

The following levels of assist are available:

Value Description

D (Display)

Activates assist mode for a particular automation flag. When Display is specified, message AOF317A is logged and added to the SDF detail status information. When you start the SDF user interface you can use assist mode to take further action. You may choose to:

- Issue the SA OS/390-generated command
- Change the command
- End the operation

Note: Do not stop the AOFTDDF task while you are using an application in assist mode as this will stop automation for your application.

Using assist mode, you can perform a step-by-step validation of commands and replies specified in the automation policy.

Attention: You should not use Assist mode before VTAM is up as you will find it impossible to respond to the assist display dialog through SDF. This will cause your automation to stall until VTAM is up.

L (Log)

Sets assist mode in logging-only mode. When an event triggers an automated action, SA OS/390 logs the action in the NetView log. The log can be reviewed to ensure that automation has run as expected.

Logging-only mode can be used to check your customized automation policy before putting it into production.

N (None)
Deactivates assist mode.

Assist mode works for all commands or replies issued using the following common routines:
- ACFCMD
- ACFREP

Assist mode works for the following generic routines as they call the ACFCMD and ACFREP common routines (if AOCQRY was called before).
- ACTIVMSG
- HALTMSG
- TERMMSG
- ISSUECMD
- ISSUEREP

When assist mode is on interactively (D), it uses SDF to display information on automation. To access SDF, type **SDF** on any NetView command line and press the ENTER key. For more information on how to use SDF see *System Automation for OS/390 User's Guide*.

When assist mode is on for a resource automation flag and an event occurs that triggers automation, the resource goes into ASSIST status and appears on the SDF panels in the color associated with the ASSIST status. (The default color for ASSIST is blinking white.)

When a resource appears in ASSIST status:
1. Move the cursor to the system or resource.
2. Press PF2.

   You see the Detail Status Display with the message from assist mode at the bottom of the panel. This message identifies the resource that is in assist mode and the command or reply that automation issues.

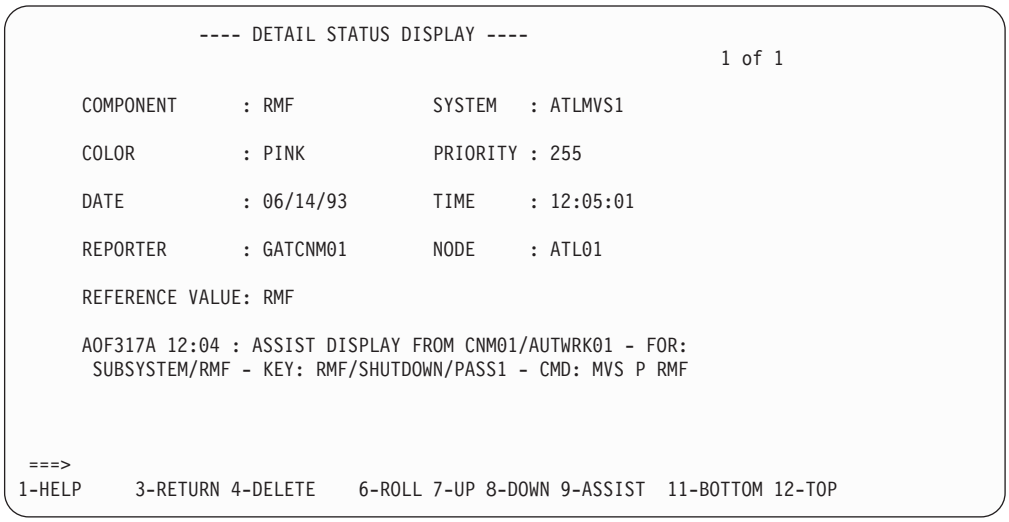


Figure 44. SDF Detail Status Display Panel with Assist Mode

In this example, the automation operator AUTWRK01 in system CNM01 issues the MVS P RMF command for the RMF subsystem. (P is an abbreviation of STOP. See the *MVS/ESA Operations: System Commands* for information on MVS commands.) This action was specified using the Shutdown option for the RMF subsystem in the customization dialog.

3. To indicate whether you want automation to continue with the command indicated, press PF9.

   You see the Operator Assist panel.

4. You now have three options. You can:
   - Route the command to the original operator with or without modifications.

     To do this, type ROUTE, and if you want to modify the command, type in your changes over the command text that is there. When you have finished modifying the command press the Enter key.
   - Delete the command from the SDF display so you are no longer in assist mode. The command is not issued and the shutdown will hang until an operator intervenes. This also changes the subsystem to the status it was in before it entered assist mode (in this case, AUTOTERM).

     Note: While the subsystem is in Assist mode, no further shutdown passes will be issued for it. This means you must action (either ISSUE or DELETE) all Assist panels from the first pass before any are generated for the second.
   - Return to the SDF without taking any action. This suspends the automation until you do take an action. (Assist mode does not time out.)

     To do this, press PF3.

   In the example, if you choose to issue the command, you type a character beside the first option. This issues the command to MVS and RMF shuts down.

Using AOCTRACE to Trace Automation Procedure Processing

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

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

The trace flag is set to null or a valid REXX trace type, which are as follows:
- A (All)
- R (Results)
- I (Intermediate)
- C (Commands)
- E (Errors)
- F (Failures)
- L (Labels)
- O (Off)
- N (Normal)

The S (Scan) trace type cannot be used.

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

AOCTRACE is documented in *System Automation for OS/390 Operator's Commands*.

REXX Coding Example

The following statements are sample code which can be placed at the beginning of your REXX automation routines to handle trace and debug settings:

```
/* REXX example of trace and debug processing */

Parse Source . . ident .
'GLOBALV GETC AOFDEBUG AOF.'||ident||'.0DEBUG AOF.'||ident||'.0TRACE'
If aofdebug = 'Y' Then
  'AOCMSG' ident||',700,LOG,'||time()||','||opid()||','||Arg(1)
loc.0debug = aof.ident.0debug
loc.0me = ident
If aof.ident.0trace <>'' Then Do
  loc.0debug = '
  Trace Value aof.ident.0trace
End
If loc.0debug = 'Y' Then
   'PIPE LIT/' ident ' called with >' Arg(1) '</' ,
   '| LOGTO NETLOG'
```

In this example, CLIST-specific debugging is disabled when the REXX tracing is activated. This is intended to reduce extraneous information which may otherwise be generated by the trace. A message is logged which shows the CLIST name, the trace setting, the operator ID and the parameters.

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

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

NetView Testing and Debugging Facilities

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

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

You can also specify less detailed tracing on the TRACE and &CONTROL statements, so that only commands are traced. A comparable facility, the interactive debugging aid, is available for programs coded in PL/1 and C.

Perform specific tracing by issuing NetView MSG LOG, PIPE LOGTO NETLOG commands at appropriate points throughout a NetView command list, REXX procedure, or PL/1 routine.

Testing and Debugging Automation Procedures

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

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

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

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

Where to Find More Testing Information

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

- *Tivoli NetView for OS/390 Customization Guide*

  This book lists requirements for your programs, including preparing your code for use, and detailed information on writing exit routines and command processors.
- *Console Automation Using NetView: Implementing*

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

Coding Your Own Information in the Automation Status File

You code your own information into the automation status file using *User E-T Pairs* in the customization dialog.

The automation status file has ten user data fields associated with each resource that is defined within it. You may use these fields to store persistent information about resources that your code needs to access later. The information in the ASF is not lost when SA OS/390 is shut down. It will last until either:

- The ASF VSAM data set is deleted and redefined, or
- You bring SA OS/390 up with an automation control file that does not include the application that the information has been defined for.

Note that you should verify that the information you have stored in the automation status file is accurate whenever SA OS/390 initializes, as circumstances may have changed while SA OS/390 was down.

Each automation status file field reserved for your data can contain up to 20 characters. The ASFUSER command allows you to update and display data in these fields. See *System Automation for OS/390 Programmer's Reference* for the ASFUSER command description.

Programming Recommendations

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

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

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

  Using NetView command list language variable JOBNAME for the resource field on an AOCQRY call, an automation procedure can be written to support a specific message for any job that can issue a message.
- Use defaults when possible to minimize coding.
- Use generic error codes (see CDEMATCH).
- Use available message parsing techniques:
  - Use the NetView command PARSEL2R or REXX PARSE command to parse a message without relying on a field position in a message.
  - Parse a message in the NetView message automation table and send only necessary fields to an automation procedure.
- Consider not coding the ENTRY field in CDEMATCH calls (default is the SUBSAPPL returned from the last AOCQRY call).
- Use appropriate automation flags.
- Review the coding requirements in *Tivoli NetView for OS/390 Customization Guide* including restrictions to consider when writing code, such as:
  - Restrictions when TVBINXIT is on
  - Variable names
  - Macro use
  - Register use
  - Re-entering programs
- Use SA OS/390 generic routines where possible, because they:
  1. Reduce your maintenance overhead.
  2. Often use internal interfaces which are more efficient than the common routines. Similarly, it is better to use a common routine than to write your own code to process the response from an ACF display request.
- Use SA OS/390's processor operations common commands where possible, because these:
  1. Are independent of the hardware type of the target system's processor
  2. Minimize the need for changes to your automation routines as you add new processors to your enterprise
- Consider using the NetView VIEW command to display online help text associated with new code, and to develop a full-screen interface for new commands that are a part of the new code. Refer to *Tivoli NetView for OS/390 Customization Guide* for information on the VIEW command.

Global Variable Names

When creating your own automation procedures, you must ensure that the names of any global variables you create do not clash with SA OS/390 external or internal global variable names. SA OS/390 external global variables are documented in *System Automation for OS/390 Defining Automation Policy*. In addition, you should not use names beginning with:

- CFG.
- AOF
- ING

- ISQ
- EVI
- EVE
- EVJ

Chapter 9. How to Automate Processor Operations Controlled Resources

This chapter contains information about how to customize your SA OS/390 installation to enable the automation of messages coming from target systems that are controlled by processor operations. These target systems or resources are referred to as ***processor operations resources*** in the following. With the method described in this chapter, you can use SA OS/390 system operations to react on these messages. This information is contained in "Automating Processor Operations Resources of OS/390 Target Systems Using Proxy Definitions", which introduces the general process how to achieve such message automation.

"Message Automation for IXC102A" on page 118 then presents a scenario how to have the special message IXC102A automated by SA OS/390.

Automating Processor Operations Resources of OS/390 Target Systems Using Proxy Definitions

SA OS/390 processor operations can be used to automate messages which cannot be automated on the target systems themselves. Typically these messages include those appearing at IPL time.

In a sysplex environment there are additional messages (XCF WTORs) being displayed at IPL time when joining the sysplex and at shutdown time when a system is leaving a sysplex. These WTOR messages cannot be automated yet because SA OS/390 system operations is not active at that time.

With the XCF message automation framework described in this chapter, you have a method of exploiting your own XCF message automation. SA OS/390 will also deliver samples in the sample policy database exploiting this framework.

Note: There are XCF WTOR messages which are automatable by Sysplex Failure Management (SFM). In these cases, to avoid conflicting automation, it is not recommended that you automate these messages by SA OS/390.

The Concept

You can use the SA OS/390 standard interface and routines to handle system external messages in almost the same way as system internally generated messages. This applies to the way of defining message automation in the customization dialog as well as to the means available for controlling message automation at automation time.

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

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

Messages which are generated on external systems, where no SA OS/390 is active or not yet active, can also be automated. Therefore, these resources generating these messages are called ***processor operations resources***. They are defined in the customization dialog as entry type *System* (SYS).

Customizing Automation for Proxy Resources

It is assumed that you have already used the customization dialog to define processor operations target systems and made these systems accessible to the processor operations focal point via the Processor Control file (see also *System Automation for OS/390 Defining Automation Policy*). So for every processor operations target system defined on the processor operations focal point, you should define a proxy resource. You do this by defining the proxy resource as entry type *Application* (APL) in the customization dialog.

Note: If you want to define many proxy resource applications, you can use the application class concept as described in *System Automation for OS/390 Defining Automation Policy*).

The rules that you need to obey when defining the proxy resource are described in the subsequent list.

Defining the proxy resource as *Application* (APL) has another advantage: The system is then visible in the INGLIST panel and it can be managed and monitored like an application resource. SA OS/390 users will be able to not only use message automation for target system messages, they also can issue start and stop commands to IPL and to shutdown systems. These commands can be defined like any start and stop command for an application. But other than application resources, target systems are managed by processor operations commands (e.g. ISQCCMD target_system_name ACTIVATE FORCE(NO) or ISQSEND target_system_name OC vary xcf,target_system_name,off,retain=yes). Processor operations commands allow you to send MVS commands to target systems as well as to send hardware commands to the processor (support element).

Here is the list of rules:

1. As mentioned before, you need to define (or have defined) the processor operations target systems that you want to automate. For those systems, the following rule applies:

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

   If this is not the case, you need to change it subsequently.
2.

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

Job Type = NONMVS
The ***Job Type*** for the proxy application must be NONMVS.

4. The ***Monitor Routine*** for the proxy application must be ISQMTSYS.
5.

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

6.

Note:
If you want to inhibit operators from performing a startup or shutdown for a target system resource using the INGREQ command, ***External Startup*** and ***External Shutdown*** must be set to 'ALWAYS'.

7. If you do not want the proxy resource to be started at an IPL or on NetView recycle of the processor operations focal point, you should specify NO for both fields ***Start on IPL*** and ***Start on RECYCLE***.
8. As you can only automate applications by linking them to systems via an application group, you need to define an application group for the proxy applications. Do not merge the proxy applications with other applications into this application group because destructive requests applied to a merged application group would also affect the proxy resources contained in that group.

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

   Enter 'cmd' in the ***Action*** column to specify the command to be processed if the defined message occurs.
10. If the message to be automated is a WTOR, then the variable &EHKVAr1 will contain the reply ID. This variable may then be used as a parameter to the ISQSEND command:

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

Startup and Shutdown Considerations

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

```
Start example:

ISQCCMD &SUBSJOB LOAD FORCE(NO)

Stop example:

Pass 1 ISQSEND &SUBSJOB OC Z EOD
```

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

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

For more details about processor operations commands refer to *System Automation for OS/390 Operator's Commands*.

Preparing Message Automation

The interaction with target systems is based on the SA OS/390 processor operations component. Therefore the installation and customization of this component must be complete at this point.

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

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

MSCOPE() parameter in CONSOLxx member
MSCOPE allows you to specify those systems in the sysplex from which this console is to receive messages not explicitly routed to this console. An asterisk (*) indicates the system on which this CONSOLE statement is defined. Since the default is *ALL, indicating that unsolicited messages from all systems in the sysplex are to be received by this console, this parameter must be set to '*' for correct automation by SA OS/390 processor operations.

Sample NetView Message Automation Table INGMSG02

This NetView message automation table fragment checks for ISQ900I and certain message IDs contained in the ISQ900I. In case of a match ISSUECMD, ACTIVMSG, TERMMSG or HALTMSG can be called with the message ID as parameter.

Message Automation for IXC102A

This chapter describes how to use SA OS/390 facilities to automate the message IXC102A which is issued by the cross-system coupling facility of the sysplex.

Note: The routine that automates the IXC102A message does not support assist mode.

Loss of Focal Point Authorization of a Support Element

The messages BNH021I and DWO345I indicate that the focal point authorization was changed from the current NetView to another one. In case the new NetView is the backup system of the current system or vice versa, no action is required. Otherwise, you have to retrieve the authority in order to automate the message IXC102A for the affected systems. You achieve this by issuing the command:

```
GETSPCFP luname
```

where the *luname* can be found in each of the messages above. If you wish to automate this operation, activate the appropriate statements in the DSIPARM member INGMSG01.

How to Customize IXC102A Automation

To enable IXC102A automation, you need to perform a series of steps from the customization dialog to edit your enterprise's policy database and build a new automation control file. These steps are described in the subsequent list.

1. From the delivered *SYSPLEX sample policy database, in the entry type *Application* in the ***Entry Type Selection*** panel, check for an application class *SYSTEM_PROXY_CLASS* and copy it into your policy database. This is an application of ***Object Type*** *CLASS* so that the policy defined for this application class can be inherited by other applications.
2. Select policy item MESSAGES/USERDATA from the application class created in step 1 and check for message *IXC102A* in the list of displayed messages and enter **cmd** into the ***Action*** column as shown in Figure 45.

```
  COMMANDS  ACTIONS  HELP
------------------------------------------------------------------------------
AOFPISSM                      Message Processing            Row 8 to 15 of 33
Command ===>                                                  SCROLL===> PAGE

Entry Type : Application           PolicyDB Name   : XCF
Entry Name : SYSTEM_PROXY_CLASS    Enterprise Name : ALL_PRIMARY

Subsystem  : SYSPROXYCLS

Enter messages issued by this resource that will result in automated actions.
Actions:  CMD = Command  REP = Reply  CODE = CODE  USER = User defined values

Action    Message ID                                  Cmd  Rep  Code  User
_______   ____________________________
          ________________________________________
cmd____   IXC102A_____________________________        4         1
          Replies DOWN when system left sysplex___
```

Figure 45. Message Processing Panel

Pressing ENTER will lead you to panel ***CMD Processing*** Figure 46 on page 120.

3. In our sample from Figure 46 on page 120, we offer four commands. For example, the command

   ```
   ISQCCMD &EHKVAR2 SYSRESET CLEAR(YES) FORCE(YES)
   ```

   will be processed when value *RESETFORCE* is used by a CDEMATCH operation.

   How to enter the values to be returned is described in step 4 on page 120.

   In the ***CMD Processing*** panel, you can change the commands or add valid ISQCCMD commands. Valid ISQCCMD commands are `LOAD`, `SYSRESET`, `ACTIVATE` or `DEACTIVATE` commands. Specifying invalid or no commands, or two or more commands will result in the processing of the default command

   ```
   SYSRESET CLEAR(NO) FORCE(YES)
   ```

   to clear pending I/O operations.

The variable &EHKVAR2 is replaced by the system name of the system that leaves the sysplex.

```
 COMMANDS  HELP
------------------------------------------------------------------------------
AOFPISSC                        CMD Processing              Row 1 to 4 of 24
Command ===>                                                 SCROLL===> PAGE

Entry Type : Application           PolicyDB Name   : XCF
Entry Name : SYSTEM_PROXY_CLASS    Enterprise Name : ALL_PRIMARY

Subsystem  : SYSPROXYCLS
Message ID : IXC102A

Enter commands to be executed when resource issues the selected message.

Pass/Selection Automated Function/'*'
Command Text
RESETCLEAR      ________
ISQCCMD &EHKVAR2 SYSRESET CLEAR(YES) FORCE(NO)_____________________________
_______________________________________________________________________________

LOADCLEAR       ________
ISQCCMD &EHKVAR2 LOAD CLEAR(YES) FORCE(NO)_____________________________
_______________________________________________________________________________

RESETFORCE      ________
ISQCCMD &EHKVAR2 SYSRESET CLEAR(YES) FORCE(YES)_____________________________
_______________________________________________________________________________

LOADFORCE       ________
ISQCCMD &EHKVAR2 LOAD CLEAR(YES) FORCE(YES)_____________________________
_______________________________________________________________________________
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP      F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 46. CMD Processing Panel

Pressing PF3 (END) will lead you back to panel ***Message Processing*** from Figure 45 on page 119.

4. You enter **code** into the ***Action*** column of panel ***Message Processing***.

   This will lead you to the ***Code Processing*** panel as shown in Figure 47.

```
 COMMANDS  HELP
------------------------------------------------------------------------------
 AOFPISSE                        Code Processing             Row 1 to 14 of 21
 Command ===>                                                  SCROLL===> PAGE

 Entry Type : Application           PolicyDB Name   : XCF
 Entry Name : SYSTEM_PROXY_CLASS    Enterprise Name : ALL_PRIMARY

 Subsystem  : SYSPROXYCLS
 Message ID : IXC102A

 Enter the value to be passed to the calling CLIST when this resource
 issues the selected message and the following codes are contained in
 the message.

     Code 1          Code 2          Code 3           Value Returned
 ISQCCMD________ _______________ _______________ RESETFORCE_____________________
 _______________ _______________ _______________ _____________________________
 _______________ _______________ _______________ _____________________________
```

Figure 47. Code Processing Panel

Do not use a value other than ***ISQCCMD*** for the ***CODE 1*** column. For the ***Value Returned*** column, use one of the selections (RESETCLEAR,

LOADCLEAR, RESETFORCE, LOADFORCE) shown in the ***CMD Processing*** panel (Figure 46 on page 120). However, in the ***Code Processing*** panel, you must not specify more than one such line.

5. Copy the *SYSTEM_PROXY* application from the *SYSPLEX policy database to your policy database and rename it to the target system name you want to automate.
6. Select policy item LINK TO CLASS from the copied application. In the upcoming ***Link Instance to Class*** panel, link this instance to *SYSTEM_PROXY_CLASS*.
7. If you want to change the predefined automation for IXC102A, then select policy item MINOR RESOURCES from either the proxy application or application class (*SYSTEM_PROXY* or *SYSTEM_PROXY_CLASS*) and select minor resource ixc102a as shown in Figure 48.

   Note: The ***Major Resource*** field from Figure 48 shows the value that has been entered into the ***Subsystem Name*** field when defining this proxy application.

```
COMMANDS  ACTIONS  HELP
------------------------------------------------------------------------------
AOFPIMR3                   Minor Resource Selection
Command ===>                                                SCROLL===> PAGE

Entry Type : Application           PolicyDB Name   : XCF
Entry Name : SYSTEM_PROXY          Enterprise Name : ALL_PRIMARY

Major Resource: SYSTEMPROXY

Select minor resource to be altered.

Action Minor Resource
s      ixc102a_____________
_      ____________________
```

Figure 48. Minor Resource Selection Panel

Pressing ENTER will bring you to panel ***Flag Automation Specification*** where you may set or unset the ***Recovery*** automation flag as desired.

```
------------------------------------------------------------------
AOFGFAS1                  Flag Automation Specification
Command ===>

Entry Type : Application           PolicyDB Name   : XCF
Entry Name : SYSTEM_PROXY          Enterprise Name : ALL_PRIMARY

Resource:  SYSTEMPROXY.IXC102A
Enter level of automation desired.

Automation Flags:  Y = Yes      N = No   E = Exits
Assist Flags:      D = Display  L = Log  N = None

 Actions    Flag         Auto      Assist
 ________  Automation .  ________  ________
 ________  Recovery . .  YES_____  ________
 ________  Start. . . .  ________  ________
 ________  ShutDown . .  ________  ________
 ________  Initstart. .  ________  ________
 ________  Restart. . .  ________  ________


 Enter or Display times to disable automation . . NO       YES     NO
```

Figure 49. Flag Automation Specification Panel (1)

Automating Linux Console Messages

The Linux Console Connection to NetView

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

Linux Console Automation with Mixed Case Character Data

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

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

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

Security Considerations

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

Restrictions and Limitations

Only Linux systems running in an LPAR or in Basic-Mode of a zSeries or 390-CMOS processor hardware are supported. Linux systems running as VM guests cannot be monitored and controlled using SA OS/390 processor operations. In the command shell environments of a Linux console it is possible to pass control keys as character strings instead of pressing the keyboard control key combination to perform functions like Control-C. The current Linux support of SA OS/390 processor operations has not been tested using this Linux capability. Any Linux program or command script that requires a user interaction with control keys should not be invoked using the SA OS/390 processor operations ISQSEND interface.

Chapter 10. SA OS/390 User Exits

To allow customer specific activities, which are not covered by the customization dialogs, SA OS/390 provides support for the following classes of user exits:

1. Static exits, which are called at fixed points in SA OS/390 processing
2. Flag exits, which are called when SA OS/390 needs to evaluate an automation flag.
3. Exits for ACF BUILD processing; exits that are called before and after the standard customization function and exits that are called for DELETE processing and COPY processing, see "Customization Dialog Exits" on page 137.
4. An exit for INGREQ processing: AOFEXC01; this is also a static exit described in "Static Exits" on page 126.

Additionally, SA OS/390 has a number of facilities which behave in an exit-like manner.

The following figure shows the sequence in which exits may be invoked during SA OS/390 initialization:

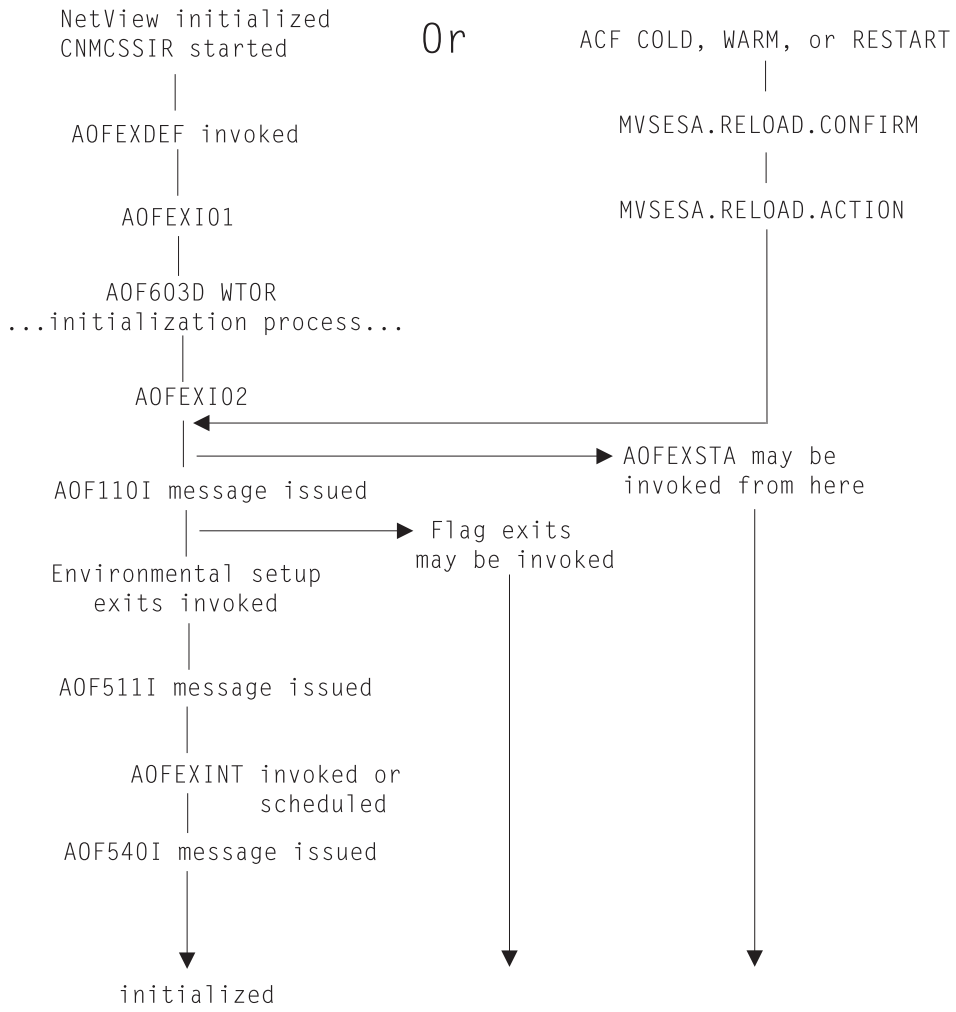


Figure 50. SA OS/390 Exit Sequence during SA OS/390 Initialization

Static Exits

These exits are invoked at fixed points in SA OS/390 processing. They are always invoked if they are found in the DSICLD concatenation. Positive return codes from these exits are generally ignored, though it is recommended that you always exit with a return code of 0.

The main purpose of static exits is to allow you to take your own actions at specific points during SA OS/390 processing. The static exits available are described below.

AOFEXDEF

This exit is called at the start of SA OS/390 initialization, before message AOF603D is issued. This exit should be used to change your advanced automation options. For example, using AOFEXDEF you can:

- Load a different MPF table.
- Set advanced automation options.

See Appendix B, "Global Variables to Enable Advanced Automation", on page 149 for information on advanced automation options.

This exit is run on AUTO1.

Parameters: None.

Return Codes: 0 is expected.

AOFEXI01

This exit is invoked before the AOF603D ENTER AUTOMATION OPTIONS reply is issued. It is invoked in a NetView PIPE and gets the data that is displayed in the AOF767I message as input in the default SAFE. With this exit you can add or remove lines from the message and add additional options to the reply.

Parameters: None.

Return Codes: 0 is expected.

AOFEXI02

This exit is invoked after the operator has replied to the AOF603D reply. It gets the operators response to the reply as input in the default safe and it can remove, add or change the options the operator has entered.

Parameters: None.

Return Codes: 0 is expected.

AOFEXI03

This exit is invoked before SA OS/390 loads message automation tables. It can be used to create statistics of the currently loaded message tables. Together with the message table listings that SA OS/390 produced at load these statistics can be used for any purpose.

Parameters: None.

Return Codes: 0 is expected.

AOFEXI04

This exit is invoked after SA OS/390 loads message automation tables. It can be used to store the message table listings that SA OS/390 produced at load.

Parameters: None.

Return Codes: 0 is expected.

AOFEXINT

This exit is called when SA OS/390 initialization is complete, before message AOF540I is issued. You can use AOFEXINT to call your own initialization processing after SA OS/390 has finished. Refer also to the description of the global variable AOFSERXINT in Appendix B, "Global Variables to Enable Advanced Automation", on page 149.

Parameters

None.

Return Codes: 0 is expected.

AOFEXSTA

This exit is called from AOCUPDT every time the automation status of an application is updated.

Note: It is not necessary for AOCUPDT to *change* an application automation status for this exit to be called. The exit is still invoked if the update does not result in a change of status.

AOFEXSTA can be used to perform any special status transition processing that cannot be triggered by other methods.

Note: This exit is invoked frequently, and will be invoked at times when SA OS/390 is not fully initialized. Your exit code should be as robust and efficient as possible.

SA OS/390 will attempt to load AOFEXSTA into storage at initialization. If this attempt fails, AOFEXSTA will not be invoked on any AOCUPDT calls. To activate the exit it must be present in the DSICLD concatenation when the automation control file is loaded or reloaded.

AOFEXSTA runs on the task that called AOCUPDT, after all other processing has finished.

Attention: AOFEXSTA is scheduled with EXCMD opid(). If your operators are issuing commands which change application statuses and you wish to use AOFEXSTA you may have to modify your scope definitions.

Parameters

Parameters are passed in sequence, delimited by commas.

Resource type
SA OS/390 uses types of SUBSYSTEM, MVSESA, WTORS, and SPOOL. Other users may use other resource types.

Resource Name
For an application, this is the name of the subsystem it is defined as.

Automation Status
For an application, this is one of the twenty six SA OS/390-supported automation statuses.

SDF Root
This is the SDF Root, as specified in the customization dialog, for the system that originated the status update. Generally the exit is driven only for status changes on other systems on the automation focal point.

Return Codes: 0 is expected.

Restrictions

- Because the exit is scheduled with EXCMD, the status update and subsequent processing in the caller will have completed before the exit is invoked.
- Check the resource type and the SDF root to ensure you are only trying to process the right things.
- Plan carefully before you take any action to change the status of an application from this exit. If you are not careful you may create a loop (AOCUPDT to AOFEXSTA to AOCUPDT to AOFEXSTA).

Note:
Consider using ISSUEREP, ISSUECMD or status change commands as alternatives to AOFEXSTA, since AOFEXSTA is invoked for **every** status update which seriously degrades performance.

The generic routines ACTIVMSG and TERMMSG will, if the advanced automation options are set up appropriately, issue commands whenever an application changes to a particular status. It may be more appropriate to place commands here, rather than in the status change exit, which gets driven for every status update of every resource. It is recommended to use status change commands for better performance.

AOFEXC00

The exit routine AOFEXC00 will be called when in the AOFPOPER panel the selection ***L*** has been entered. No parameters are passed to the routine. The purpose of this routine is to act as the starting point for installation provided local functions.

AOFEXC01

If this exit routine is defined, it will be invoked during INGREQ processing after the verification panel is displayed and the caller replied to continue. The following parameters are passed from the INGREQ command and are positional:

Parameters:

request
is the request type

SCOPE
The scope of the command

OVERRIDE
is the override specification

RESTART
is the restart specification

TYPE contains the start/stop type

PRIORITY
is the priority given to the request

SOURCE
identifies the originator of the request

REMOVE
indicates the condition under which the request is automatically removed

TO_INTERVAL
specifies the timeout interval

TO_OPTION
specifies the timeout option

EXPIRATION
specifies the date and time when the request is removed

APPL_PARMS
specifies the application parameters

COMMENT
is the comment - given by the operator - associated with the request

Note: The parameters are separated by a comma.

Return Codes:

0 OK - continue

1 error - reject command

The list of resources involved in the INGREQ command is passed to the exit by using the default SAFE. Each resource is described by its location and name. The format of the location is:

```
sysplex_name.domain_ID.system_name\sa_version\xcf_groupname
```

Specifying the xcf_groupname is optional.

The format of the resource name is:

```
name/type/system_name
```

For example:

```
AOCPLEX.IPUFA.SA1D\V2R1M0 RMFGAT/APL/SA1D
```

The user exit is called in a PIPE. If the user exit returns a bad RC and additional data is written to the console, this data is shown in a message panel. If no additional data is passed in the exit, then message AOF227 is issued.

AOFEXC02

If this exit routine is defined, it is invoked during INGSCHED processing before the schedule override file is updated. The parameters are positional and separated by a comma. The following parameters are passed to the exit:

Parameters:

user ID
is the user ID making the update or delete

resource
The resource is described by two words. The first word is the location of the resource. The second word is the resource name. The format of the location is:

```
sysplex_name.domain_ID.system_name\sa_version
```

The format of the resource name is:

```
name/type/system_name
```

For example:

```
AOCPLEX.IPUFA.AOC8\V2R1M0 TSO/APL/AOC8
```

action can be UPD or DEL

date specifies the date in the format YYYYMMDD

UP priority
specifies the priority. It can be L or H

UP time slots
specifies the time slots when the application is up. The format is hhmm-hhmm... hhmm-hhmm

DOWN priority
specifies the priority.

DOWN
specifies the time slots when the application is down. The format is hhmm-hhmm... hhmm-hhmm

Return Codes:

0 OK - continue

1 error - reject command

The user exit is called in a PIPE. If the user exit returns a bad return code and additional data is written to the console, this data is shown in a message panel. If no additional data is passed in the exit, then message AOF227 is issued.

The format of the location is:

```
sysplex_name.domain_ID.system_name\sa_version\xcf_groupname
```

Specifying the xcf_groupname is optional.

AOFEXC03

If this exit routine is defined, it is invoked by the DISPINFO command slave to retrieve user-supplied information about the subsystem. The input for the routine is the subsystem name. The data returned by the exit is shown as part of the DISPINFO output.

Note: Certain special symbols are interpreted as panel attribute symbols. For more information about attribute symbols, refer to the *NetView Customization Guide*. The DISPINFO panel uses the default attribute set 1. This allows the exit to color the data to be displayed. To display a particular symbol, place a double quotation mark (“) in front of the character.

Parameters:

subsystem name
Is the name of the subsystem.

Return Codes:

0 OK.

1 An error occurred.

AOFEXC04

If this exit routine is defined, the command code U is supported for the DISPSTAT and INGLIST commands. The input for the AOFEXC04 exit is the resource name (subsystem name for DISPSTAT) and the location of the resource. The location is either the system name if the resource resides on a system member of the local sysplex, or the domain ID if the resource resides on a system which is outside of the local sysplex.The parameters are separated by a comma.

Parameters:

subsystem name
Is the location of the subsystem. It is either the system name if the subsystem resides on a system member of the local sysplex, or the domain ID if the subsystem resides on a system which is outside of the local sysplex.

Environmental Setup Exits

The SA OS/390 customization dialog allows you to define a string of exits which are invoked during SA OS/390 initialization processing. These exits are defined using the AUTOMATION SETUP policy item of the *System* policy object. See *System Automation for OS/390 Defining Automation Policy* for more information. Environmental setup exits are invoked after SA OS/390 has started its various tasks, but before the primary automation table has been loaded. You can use these exits to initiate your own automation, but some SA OS/390 services may be unavailable as SA OS/390 has not yet finished initializing when these exits are called. In particular, status information may be inaccurate as SA OS/390 may not have finished resynchronization. Environmental setup exits runs on AUTO1.

Parameters

Parameters are passed in sequence, delimited by blanks.

INITIALIZATION
INITIALIZATION is a constant.

Either RELOAD or REFRESH or IPL or RECYCLE

RELOAD indicates that the automation control file has been reloaded.
REFRESH indicates that the automation control file has been refreshed.
IPL indicates that SA OS/390 has just been restarted after a system IPL.
RECYCLE indicates that NetView has been restarted.

Return Codes

0 is expected. If you return a non-zero return code you may prevent other exits from being invoked or disrupt SA OS/390 initialization.

Usage Notes

- These exits are not driven if you run RESYNC.
- Unlike the other static exits, you must specify the name of the routine or routines to invoke in the automation control file.

Flag Exits

Using automation flag exits you can cause your automated operations code to exit normal SA OS/390 processing to an external source, such as a scheduling function, to determine whether automation should be on or off for a given resource at that particular instant.

Flag exits can be defined for :

- Any flag (Automation, Initstart, Start, Recovery, Shutdown or Restart).
- Any resource.
- Any minor resource. See the description of the policy item MINOR RESOURCES in *System Automation for OS/390 Defining Automation Policy* for more information on minor resources.

You can specify multiple exits for each flag. A flag exit is invoked only if SA OS/390 needs an "opinion" on the current flag setting. Flag exits and flags all work on a "veto" basis. A flag is ON when all flags and flag exits agree that it is on.

Flags are set to ON, OFF, or EXIT.

- When a flag is set ON, exits are not called.
- When a flag is set OFF, exits are not called.
- When a flag is set EXIT, the exits are checked.

Specifying FORCE=YES on your AOCQRY call will force the exits to be called when the flag is set to ON or OFF. In this case a flag exit can turn an ON flag OFF, but it cannot turn an OFF flag ON.

Flag settings are determined by:

- The automation control file
- NOAUTO periods (the flag is OFF during a NOAUTO period)
- User-entered INGAUTO and TIMEAUTO commands.

For example, if the following flag settings are entered:

```
Resource      Flag         Setting
------------  ------------ --------------------------
DEFAULTS      AUTOMATION   ON
SUBSYSTEM     RESTART      OFF
JES2          AUTOMATION   Call Exit J2AUT
JES2          START        Call Exit J2STR
JES2          SHUTDOWN     Call Exits J2SD1 and J2SD2
JES2          RECOVERY     OFF
```

the effective flags for JES2 are:

```
Flag          Effective setting
------------  ------------------------------------
AUTOMATION    Call Exit J2AUT
INITSTART     ON
START         Call Exit J2STR
RECOVERY      OFF
SHUTDOWN      Call Exits J2SD1 and J2SD2
RESTART       OFF
```

When SA OS/390 checks the current value of any flag for the JES2 application, the process is as follows:

```
AUTOMATION - Call Exit J2AUT
             If OFF,
                AUTOMATION flag is OFF
             If ON,
                AUTOMATION flag is ON
INITSTART  - Call Exit J2AUT
             If OFF,
                INITSTART flag is OFF
             If ON,
                INITSTART flag in ON
START      - Call Exit J2AUT
             If OFF,
                START flag is OFF
             If ON,
                Call Exit J2STR
                If OFF,
                   START flag is OFF
                If ON,
                   START flag is ON
RECOVERY   - The RECOVERY flag is OFF
SHUTDOWN   - Call Exit J2AUT
             If OFF,
                SHUTDOWN flag is OFF
             If ON,
                Call Exit J2SD1
                If OFF,
                   SHUTDOWN flag is OFF
                If ON,
                  Call Exit J2SD2
                  If OFF,
                     SHUTDOWN flag is OFF
                  If ON,
                     SHUTDOWN flag is ON
RESTART    - The RESTART flag is OFF
```

Notes:

1. No exit is called for the Recovery and Restart flags. This is because the specific flags have been turned off. The Recovery flag is OFF at the application level. The Restart flag is OFF at the application defaults (SUBSYSTEM) level. The exits cannot change the state of these flags, so SA OS/390 does not invoke them.
2. The J2STR and J2SD1 exits are called only if the J2AUT exits indicate that automation is allowed.
3. The J2SD2 exit is called only if the J2SD1 exits indicate that automation is allowed. The J2SD1 exit is called only if the J2AUT exit indicates that automation is allowed.

As this example shows, you should not assume that an exit is called every time SA OS/390 needs to evaluate the flag that it is defined on. You *can* assume that an exit is called before SA OS/390 decides that a given flag is ON and takes action on the basis of the flag setting. Additionally, you should think carefully about initiating processes from within flag exits as a later exit may give a return code which will indicate that the flag is turned OFF.

Parameters

Parameters are supplied in sequence, delimited by blanks.

Flag
This is the name of the flag that is being checked. Possible values are Automation, Initstart, Start, Recovery, Terminate or Restart.

Note: The Terminate flag is referred to as the Shutdown flag elsewhere.

Time Setting
Time Setting is a constant. It can be either:

- AUTO - automation is currently turned on.
- NOAUTO - automation is currently turned off by a NOAUTO period.

A value of NOAUTO is possible only if AOCQRY is called with FORCE=YES specified.

Note: This ensures that the exit is invoked, but it is not possible for an exit to override a NOAUTO period.

Resource Name
This is the name of the resource that the flag is being checked for. For minor resources it will contain the fully qualified minor resource name that the exit has been defined for. Given no definition for TSO.USER.MAG1 and an exit defined for TSO.USER, the resource name passed to the exit would be TSO.USER if a check was made for TSO.USER.MAG1.

This behavior is slightly different for exits that are defined for DEFAULTS or SUBSYSTEM. In this case the resource name passed is the name of the application that the flag is being evaluated for. Given no definition for TSO AUTOMATION and an exit defined for SUBSYSTEM AUTOMATION, the exit is invoked with a resource name of TSO.

Resource Type
This is always SUBSYSTEM, regardless of the actual type of the resource.

Target Prefix
This is the TGPFX value with which AOCQRY was invoked. If TGPFX is not specified, the value SUB is passed.

Return Codes

0 Automation is allowed by the exit.

greater than 0
Automation is not allowed.

Attention: You should not return a return code of -5 as this will cause multiple CLIST abends (AOFRAEXI, AOFRSCHK, caller, and others) and may seriously disrupt automation. Symptoms of this are AOF760 messages for the SA OS/390 CLIST that invoked the exit, for any CLIST that invoked the SA OS/390 CLIST, to the initiating CLIST.

Notes:

1. Flag exits are always called through AOCQRY. This means that the TGLOBALS for the application have been primed and are available for use. Normally the set of globals are found in the SUB task globals, but if AOCQRY is called with TGPFX then they will be in a different set. You should use the TGPFX parameter that is passed to locate the globals.
2. AOCQRY can be invoked in a manner that will determine a flag value but not set up the globals. You should be careful if you write code which invokes AOCQRY in this way and you have exits which rely on the task globals being set up.
3. Your exit should not assume that AOCQRY has been called without TGPFX being specified. That is, do not assume that the SUB task globals refer to the resource it has been invoked for.

4. If an exit is invoked for a minor resource, the task globals are set for the major resource associated with that minor resource.
5. If an exit is invoked for a non-subsystem resource, most of the task globals will be meaningless.
6. If you call AOCQRY from inside your exit you must specify a TGPFX value. You cannot use SUB. The TGPFX value you specify should be different from the TGPFX parameter you were passed. You are responsible for ensuring the uniqueness of all TGPFXs if you nest AOCQRY exits. Since this can become quite complex, it is recommended you avoid nesting exits.
7. Do not code calls to ACFCMD, ACFREP, or CDEMATCH as these use the SUB task globals, which may not be set up for the application that you want to process.
8. Do not change any of the AOCQRY task globals.
9. Flag exits may be called frequently, so performance is important.
10. If AOCQRY is specified with FORCE and multiple exits are defined for a flag, the exits are called in order. If an exit indicates that the flag is OFF, subsequent exits will not be called.

Pseudo-Exits

This section discusses a number of places where SA OS/390 either makes special use of a flag exit or has a function with certain, exit-like, qualities.

Automation Control File Reload Permission Exit

When an operator asks SA OS/390 to reload the automation control file, SA OS/390 checks the automation flag of minor resource MVSESA.RELOAD.CONFIRM. If the flag is turned OFF, the automation control file reload is not allowed. If the flag is turned ON, the task global AOFCONFIRM is checked. If AOFCONFIRM has been set to a non-null value, the user is prompted to confirm that they want the automation control file to be reloaded.

Notes:

1. Note that an exit can be associated with the automation flag for this resource.
2. An automation control file cannot be loaded if the automation flag for major resource MVSESA is set to 'N'. If the automation flag for minor resource MVSESA.RELOAD.CONFIRM is set to 'Y', reload of the ACF is permitted.

Automation Control File Reload Action Exit

After the automation control file reload permission exit is checked, when SA OS/390 is committed to reloading the automation control file, it will check the automation flag for minor resource MVSESA.RELOAD.ACTION. The actual setting of this flag (ON or OFF) is ignored, but any exits defined for it are invoked. All exits should return a return code of 0.

Subsystem Up at Initialization Commands

Using the customization dialog you can specify commands that are run if SA OS/390 finishes resynchronizing statuses and an application is found to be up. These commands can be useful for synchronizing local automation that has been built on top of SA OS/390.

Testing Exits

Exits should be well tested with a variety of different input parameters before they are put into production. For exits that need AOCQRY task globals, you can call AOCQRY to set up the globals without evaluating the flag exits, and then invoke the exit on its own for testing purposes. This method saves the overhead of calling AOCQRY every time you run the exit.

Attention!
If you have a syntax error or a no-value-condition in your exit it can cause parts of SA OS/390 to abend, resulting in severe disruption of your automation.

Customization Dialog Exits

SA OS/390 provides a series of user exits that can be invoked during certain phases while working with the customization dialog. They are:

- "User Exits for BUILD Processing"
- "User Exits for COPY Processing" on page 138
- "User Exits for DELETE Processing" on page 139
- "User Exits for CONVERT Processing" on page 139

"Invocation of Customization Dialog Exits" on page 140 provides information on how to activate the user exits.

User Exits for BUILD Processing

The following user exits are provided for the process of building the automation control file (BUILDF).

- INGEX10, which is called before the automation control file build function starts. This exit is only available when the build process is initiated from the customization dialogs.
- INGEX01, which is called before the automation control file build function starts. starts. This exit is available when the build process is initiated from the customization dialogs, from a batch job submitted via the customization dialogs, or from a batch job submitted independently from the customization dialogs.

  When a BUILD mode of BATCH is selected in the customization dialogs, the JCL for the batch job is submitted and INGEX01 is called when the job begins execution and before the automation control file build function starts in batch.
- INGEX02, which is called after the automation control file build function (BUILDF) has ended. This exit is available when the BUILD process is initiated from the customization dialogs, from a batch job submitted through the customization dialogs, or from a batch job submitted independently from the customization dialogs.

The following parameters are passed to both INGEX01 and INGEX02 exits, separated by commas:

- Parm1 = PolicyDB name
- Parm2 = Enterprise name
- Parm3 = BUILD output data set
- Parm4 = entry type (or blank)

- Parm5 = entry name (or blank)
- Parm6 = BUILD type (MOD/ALL)
- Parm7 = BUILD mode (ONLINE/BATCH)
- Parm8 = Configuration (0=NORMAL/1=ALTERNATE)
- Parm9 = Sysplex name (or blank)
- Parm10 = Build option (1,2, or 3)

If user exit INGEX10 produces return code RC = 0, the BUILDF processing continues. If a return code RC > 0 is produced, an error message is returned and the BUILDF processing terminates.

If user exit INGEX10 ends with return code RC > 0, user exits INGEX01 and INGEX02 are not called. Processing terminates.

If user exit INGEX10 ends with return code RC > 0 and a BUILD mode of BATCH was selected in the customization dialogs, no JCL is submitted to run the build in batch (because BUILDF does not start). Processing terminates.

If user exit INGEX01 produces return code RC = 0, the BUILDF processing continues. If a return code RC > 0 is produced, an error message is returned. BUILDF processing terminates. If the build is run in batch mode, and a return code RC > 0 is produced, the job completes with a return code RC 08.

If user exit INGEX01 ended with return code RC > 0, user exit INGEX02 are not (because BUILDF does not start). Processing terminates.

User exit INGEX02 is always called when the BUILDF process has started, irrespective of whether it has completed or not.

If user exit INGEX02 produces a return code RC > 0, an error message is displayed. If the build is run in batch mode, and a return code RC > 0 is produced, the job completes with a return code RC 04. If a severe build error occurred, the job completes with a return code RC 20.

User Exits for COPY Processing

Two user exits are implemented for the COPY processing.

1. INGEX03, which is called before the COPY function starts. The following parameters are passed:
   - Entry name of the entry to be copied to (target)
   - Entry name of the entry to be copied from (source)
   - Entry type (e.g. APL)
2. INGEX04, which is called after the COPY function has ended. The following parameters are passed:
   - Entry name of the entry to be copied to (target)
   - Entry name of the entry to be copied from (source)
   - Entry type (e.g. APL)
   - Indicator whether the COPY process was successful or not (S=successful, U=unsuccessful)

If user exit INGEX03 produces return code RC = 0, the COPY processing continues. If a return code RC > 0 is produced, an error message is displayed, the COPY function will not start, and processing terminates.

If user exit INGEX03 ended with return code RC > 0, the user exit INGEX04 will not be called as the COPY processing will terminate.

User exit INGEX04 is always called once the COPY function has started. The information about the success or failure of the COPY function is passed as a parameter.

If user exit INGEX04 produces a return code RC > 0, an error message is displayed.

User Exits for DELETE Processing

Two user exits are implemented for the DELETE processing.

1. INGEX05, which is called before the DELETE process starts. The following parameters are passed:
   - Entry name of the entry to be deleted
   - Entry type (e.g. APL)
2. INGEX06, which is called after the DELETE process has ended. The following parameters are passed:
   - Entry name of the entry to be deleted
   - Entry type (e.g. APL)
   - Indicator whether the DELETE process was successful or not (S=successful, U=unsuccessful)

If user exit INGEX05 produces return code RC = 0, the DELETE processing continues. If a return code RC > 0 is produced, an error message is displayed, the DELETE function will not start and the processing terminates.

If user exit INGEX05 ended with a return code RC > 0, user exit INGEX06 will not be called as the DELETE processing will terminate.

User exit INGEX06 will always be called once the DELETE function has started. The information about the success or failure of the DELETE function will be passed as a parameter.

If user exit INGEX06 produces a return code RC > 0, an error message will be displayed.

User Exits for CONVERT Processing

Two user exits are implemented for the CONVERT processing.

1. INGEX07, which is called before the CONVERT process starts. No parameters are passed.
2. INGEX08, which is called after the CONVERT process has ended. No parameters are passed.

If user exit INGEX07 produces return code RC = 0, the CONVERT processing continues. If a return code RC > 0 is produced, an error message is displayed, the CONVERT function will not start and the processing terminates.

If user exit INGEX07 ended with a return code RC > 0, user exit INGEX08 will not be called as the CONVERT processing will terminate.

User exit INGEX08 will always be called once the CONVERT function has started.

If user exit INGEX08 produces a return code RC > 0, an error message will be displayed.

Invocation of Customization Dialog Exits

The user exits are part of the SA OS/390 product. Therefore they are supplied in the same data set as all other ISPF REXX modules (part of SINGIREX). The supplied samples for ACF BUILD, DELETE, and COPY processing just do a 'RETURN' with return code RC=0.

You have two possibilities to apply your user modifications:

1. Edit the user exit(s) in the supplied library. Your changes will not have any consequences on the code of the SA OS/390, product. These exits will not be serviced (via PTF) by IBM® as they do not include any code at the time of product delivery.
2. Supply the modified user exit in a private data set. Then you have to concatenate your private data set to your SYSEXEC library chain. As INGDLG supports multiple data set names specified for ddname SYSEXEC, this can be done in the following way:

   ```
   INGDLG SELECT(ADMIN) ALLOCATE(YES) HLQ(SYS1)
          SYSEXEC(usr.private.dsn SYS1.SINGIREX)
   ```

   This example assumes that the high level qualifier of the data sets where the IBM supplied parts exist is SYS1.

   If you specify the SYSEXEC parameter in the INGDLG call, you need to specify the IBM supplied library explicitly with its **fully qualified** data set name.

Appendix A. Table of External Global Variables

Note: You must ensure that the names of any global variables you create do not clash with SA OS/390 external or internal global variable names. You should check this table before creating any global variables of your own. In addition, you should not use names beginning with EHK, CFG., AOF, ING, EVE, EVI, or EVJ.

Table 4. Externalized Common Global Variables

Variable Name	Description	Class	Reference
AOF.*clist*.0DEBUG	Contains either a Y or blank. If it contains Y then an intermediate level of debug supported by SA OS/390 CLISTs is turned on.	2	
AOF.*clist*.0TRACE	Contains a REXX trace setting to be used by the CLIST *clist*.	2	
AOFACFINIT	Depending on the setting, SA OS/390 will attempt to proceed with initialization despite error messages such as AOF722I during the processing of the automation control file.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFAOCCLONE*n*	Where *n* either does not exist (AOFAOCCLONE) or is a value from 1 to 9. The AOFAOCCLONE global variables contain the values specified for the &AOCCLONE. IDs for this system.	1	See the description of the *System* policy object in *System Automation for OS/390 Defining Automation Policy*.
AOF_ASSIGN_JOBNAME	This indicates whether SA OS/390 will exploit Tivoli NetView 1.2 ″ASSIGN BY JOBNAME″ feature.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFCNMASK	The characters which are used in determining unique console names.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFCTLOPT	This indicates that SA OS/390 will ignore any IPL or RECYCLE options defined for a subsystem that has a status of CTLDOWN at SA OS/390 initialization.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFDEBUG	Contains a REXX trace setting to be used globally.	2	See *System Automation for OS/390 Planning and Installation*.
AOFDEFAULT_TARGET	sets a default for the TARGET parameter for all commands where this parameter is used.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOF_EMCS_AUTOTASK_ ASSIGNMENT	Determines whether SA OS/390 will assign autotasks to EMCS consoles with a status of MASTER or ACTIVE.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOF_EMCS_CN_ ASSIGNMENT	Determines whether SA OS/390 will obtain EMCS consoles with unique names for OSTs.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.

Table of External Global Variables

Table 4. Externalized Common Global Variables (continued)

Variable Name	Description	Class	Reference
AOFEXPLAIN_USER	The EXPLAIN command accepts this variable to include help support for customer installation supplied terms. It can hold one or more pairs of ***term/help panel*** specifications separated by a blank. If the specified status in the EXPLAIN command is not a valid SA OS/390 status, the command routine will check whether it is an installation defined term. If so, the associated help panel is displayed.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFEXPLAIN_USER	The EXPLAIN command accepts this variable to include help support for customer installation supplied terms. It can hold one or more pairs of ***term/help panel*** specifications separated by a blank. If the specified status in the EXPLAIN command is not a valid SA OS/390 status, the command routine will check whether it is an installation defined term. If so, the associated help panel is displayed.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOF_GATOPER_PW_CHANGE	This variable allows you to specify if the password for the Gateway operator will be changed every 30 days or if it should keep the same password all the time.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFINITIALSTARTTYP	Contains the value 'IPL' or 'RECYCLE' depending on whether SA OS/390 has been started the first time after an IPL or after a NetView recycle.	1	
AOF_INIT_MCSFLAG	This variable contains the MCSFLAG that is used for WTOs and WTORs that are issued by SA OS/390 during initialization.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOF_INIT_ROUTCDE	This variable contains the ROUTCDE (routing code) that is used for WTOs and WTORs that are issued by SA OS/390 during initialization.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOF_INIT_SYSCONID	This variable contains the SYSCONID that is used for WTOs and WTORs that are issued by SA OS/390 during initialization.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOF_NETWORK_ DOMAIN _ID	Contains the domain name for the NetView that runs network automation as defined in the customization dialog. If not defined, the value of this variable is null.	1	See the description of the *System* policy object in *System Automation for OS/390 Defining Automation Policy*.

Table 4. Externalized Common Global Variables (continued)

Variable Name	Description	Class	Reference
AOF_PRODLVL	Contains the release level of AOC/MVS or SA OS/390. • For SA/MVS Release 2 the value is SA/MVS, V1R2M0. • For SA OS/390 1.3, the value is SA OS/390, V1R3M0. • For SA OS/390 2.1, the value is SA OS/390, V2R1M0. • For SA OS/390 2.1, the value is SA OS/390, V2R2M0.	1	
AOFINITREPLY	Allows you to skip the initial reply AOF603D.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFJESPREFX	The command prefix for the primary scheduling subsystem.	1	
AOFLOCALHOLD	Determines if the SETHOLD command will be executed automatically or if it must be manually invoked.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFMINORCHK	Indicates whether the automation settings for minor resources should be checked when automated commands are about to be issued in response to either a status change or a message.	3	See "Minor Resources" in *System Automation for OS/390 Defining Automation Policy*, also see Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFMOVOPT	The setting of this variable will indicate whether or not SA OS/390 is to ignore any IPL or RECYCLE options defined for a subsystem that has a status of MOVED at initialization. This does not apply to any subsystem that is defined to Automatic Restart Manager that SA OS/390 is aware of.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFPAUSE	The number of seconds that SA OS/390 will allow for subsystems that have shut down to be cleared by MVS, in addition to their termination delay.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFQUICKWTOR	Do not trust the CGLOBALS for WTOR reply processing.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFRELOADOPT	Will determine the action to be taken with inactive subsystems during an ACF reload.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFRESTARTALWAYS	Determines whether a subsystem that has been shut down normally, outside the control of SA OS/390, will be restarted regardless of whether or not it has reached its critical error threshold.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFRPCWAIT	This is the wait time (in seconds) that SA OS/390 will wait for command responses from other systems in the sysplex.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.

Table of External Global Variables

Table 4. Externalized Common Global Variables (continued)

Variable Name	Description	Class	Reference
AOFSENDGMFHSREQUEST	Determines whether alert forwarding will be made from a target system to the NMC focal point.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFSERXINT	Determines how the exit AOFEXINT is processed.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFSETSTATEOVERRIDE	Sets the default OVERRIDE value for the SETSTATE command.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFSETSTATESCOPE	allows you to override the predefined default for the SCOPE parameter of the SETSTATE command as documented in *System Automation for OS/390 Operator's Commands*.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFSETSTATESTART	allows you to override the predefined default for the START parameter of the SETSTATE command as documented in *System Automation for OS/390 Operator's Commands*.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOF_SET_AVM_ RESTART_EXIT	SA OS/390 will set the AVM restart exit during the initialization of the automation environment.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFSHUTCHK	Sets the default PRECHECK parameter for the SHUTSYS command.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFSHUTDELAY	Determines the number of minutes that SA OS/390 will wait for a termination message before continuing the shutdown process.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFSHUTOVERRIDE	Will set the default OVERRIDE value for the INGREQ command.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFSHUTSCOPE	Sets the default SCOPE parameter for the SHUTSYS command.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFSPOOLFULLCMD	Determines whether SA OS/390 is to issue Spool Full recovery passes more than once.	3	See the description of the *Application* policy object in *System Automation for OS/390 Defining Automation Policy*.
AOFSPOOLSHORTCMD	Determines whether SA OS/390 is to issue Spool Short recovery passes more than once.	3	See the description of the *Application* policy object in *System Automation for OS/390 Defining Automation Policy*.
AOF3WTIME	This is the number of seconds that the SHOWME command will wait for command responses.	3	See Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
AOFSUBSYS	The subsystem name of the primary scheduling subsystem.	1	
AOFSYSNAME	Contains the name of the system.	1	See AOCUPDT in *System Automation for OS/390 Programmer's Reference*.

Table 4. Externalized Common Global Variables (continued)

Variable Name	Description	Class	Reference
AOFSYSTEM	Contains the system type (MVSESA) as defined in the customization dialog.	1	The ***Environment Setup*** panel of the customization dialog.
DISPEVT_WAIT	Sets the WAIT parameter of the DISPEVT command to the specified value.	3	See DISPEVT in *System Automation for OS/390 Operator's Commands*.
DISPEVTS_WAIT	Sets the WAIT parameter of the DISPEVTS command to the specified value.	3	See DISPEVTS in *System Automation for OS/390 Operator's Commands*.
DISPTRG_WAIT	Sets the WAIT parameter of the DISPTRG command to the specified value.	3	See DISPTRG in *System Automation for OS/390 Operator's Commands*.
INGAUTO_INTERVAL	Sets the default for the INTERVAL parameter of the INGAUTO command.	3	See INGAUTO in *System Automation for OS/390 Operator's Commands*.
INGEVENT_WAIT	Sets the WAIT parameter of the INGEVENT command to the specified value. The parameter specifies whether or not to wait until the request is complete.	3	See INGEVENT in *System Automation for OS/390 Operator's Commands*.
INGGROUP_WAIT	Sets the WAIT parameter of the INGGROUP command to the specified value. The parameter specifies whether or not to wait until the request is complete.	3	See INGGROUP in *System Automation for OS/390 Operator's Commands*.
INGINFO_WAIT	Sets the WAIT parameter of the INGINFO command to the specified value.	3	See INGINFO in *System Automation for OS/390 Operator's Commands*.
INGLIST_WAIT	Sets the WAIT parameter of the INGLIST command to the specified value.	3	See INGLIST in *System Automation for OS/390 Operator's Commands*.
INGRELS_WAIT	Sets the WAIT parameter of the INGRELS command to the specified value.	3	See INGRELS in *System Automation for OS/390 Operator's Commands*.
INGREQ_ORIGINATOR	Assigns individual originators for each operator issuing an INGREQ command. If the global is set, the originator ID consists of the string 'OPER_' followed by the operator ID. This makes the originator ID unique. The result is that an INGREQ command issued by operator A does no longer replace a previous INGREQ command for the same resource issued by operator B. By default all operators are grouped together under the originator ID OPERATOR.	3	See INGREQ in *System Automation for OS/390 Operator's Commands*.
INGREQ_OVERRIDE	Sets the default OVERRIDE parameter of the INGREQ command to the specified value.	3	See INGREQ in *System Automation for OS/390 Operator's Commands*.
INGREQ_PRECHECK	Sets the default PRECHECK parameter of the INGREQ command to the specified value.	3	See INGREQ in *System Automation for OS/390 Operator's Commands*.
INGREQ_PRI	Sets the default priority (PRI parameter) of the INGREQ command to the specified value.	3	See INGREQ in *System Automation for OS/390 Operator's Commands*.

Table of External Global Variables

Table 4. Externalized Common Global Variables (continued)

Variable Name	Description	Class	Reference
INGREQ_REMOVE	Sets the default value for the REMOVE parameter of the INGREQ command to the specified value. It the resource reaches the specified status (condition), the request is automatically removed.	3	See INGREQ in *System Automation for OS/390 Operator's Commands*.
INGREQ_RESTART	Sets the default for the RESTART parameter of the INGREQ command when shutting down the resource.	3	See INGREQ in *System Automation for OS/390 Operator's Commands*.
INGREQ_SCOPE	Sets the SCOPE parameter of the INGREQ command to the specified value.	3	See INGREQ in *System Automation for OS/390 Operator's Commands*.
INGREQ_TIMEOUT	Sets the interval in minutes used to check for the INGREQ command used to check whether the request has been successfully completed, and whether to send a message or cancel the request if it has not been satisfied after that time.	3	See INGREQ in *System Automation for OS/390 Operator's Commands*.
INGREQ_TYPE	Sets the default startup/shutdown type (TYPE parameter) of the INGREQ command to the specified value.	3	See INGREQ in *System Automation for OS/390 Operator's Commands*.
INGREQ_VERIFY	Sets the default VERIFY parameter of the INGREQ command to the specified value.	3	See INGREQ in *System Automation for OS/390 Operator's Commands*.
INGREQ_WAIT	Sets the WAIT parameter of the INGREQ command to the specified value.	3	See INGREQ in *System Automation for OS/390 Operator's Commands*.
INGSCHED_WAIT	Sets the WAIT parameter of the INGSCHED command to the specified value. The parameter specifies whether or not to wait until the request is complete.	3	See INGSCHED in *System Automation for OS/390 Operator's Commands*.
INGSET_VERIFY	Sets the default VERIFY parameter of the INGSET command to the specified value.	3	See INGSET in *System Automation for OS/390 Operator's Commands* and Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
INGSET_WAIT	Sets the WAIT parameter of the INGSET command to the specified value. The parameter specifies whether or not to wait until the request is complete.	3	See INGSET in *System Automation for OS/390 Operator's Commands*.
INGTRIG_WAIT	Sets the WAIT parameter of the INGTRIG command to the specified value.	3	See INGTRIG in *System Automation for OS/390 Operator's Commands*.
INGVOTE_EXCLUDE	Sets the EXCLUDE parameter of the INGVOTE command to the specified value.	3	See INGVOTE in *System Automation for OS/390 Operator's Commands* and Appendix B, "Global Variables to Enable Advanced Automation", on page 149.

Table 4. Externalized Common Global Variables (continued)

Variable Name	Description	Class	Reference
INGVOTE_STATUS	Sets the STATUS parameter of the INGVOTE command to the specified value. The parameter specifies which requests should be displayed - winning, losing or all.	3	See INGVOTE in *System Automation for OS/390 Operator's Commands* and Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
INGVOTE_VERIFY	Sets the default VERIFY parameter of the INGVOTE command to the specified value.	3	See INGVOTE in *System Automation for OS/390 Operator's Commands* and Appendix B, "Global Variables to Enable Advanced Automation", on page 149.
INGVOTE_WAIT	Sets the WAIT parameter of the INGVOTE command to the specified value.	3	See INGVOTE in *System Automation for OS/390 Operator's Commands*.
WAITTIME	The time SA OS/390 waits for a response after issuing a command before an error condition is raised. Defined in the customization dialog.	1	See the description of the *Timeout Settings* policy object in *System Automation for OS/390 Defining Automation Policy*.
XDOMTIME	The time that message forwarding routines wait for a response before it is assumed that the logon attempt failed. Defined in the customization dialog.	1	See the description of the *Timeout Settings* policy object in *System Automation for OS/390 Defining Automation Policy*.

Note: There are three different classes of variables in this table, based on the level of access available to the programmer:

- Class 1: Read-only variables. These variables are set by SA OS/390 and require at minimum an automation control file reload to be changed.
- Class 2: Read-only variables. These variables are set by SA OS/390 CLISTs. They should not be changed except by calling the appropriate CLISTs.
- Class 3: Read/write variables. These variables may be set by your own code. It is recommended that you use the ***AOFEXDEF*** exit to assign a value to these CGLOBALS.

Table of External Global Variables

Appendix B. Global Variables to Enable Advanced Automation

Table 5 lists the common global variables that you can set in your startup exit to change the way that SA OS/390 behaves. These variables should be set only once for an SA OS/390 system. You can enable or disable advanced automation options by changing the settings of the global variables in your initialization defaults exit (AOFEXDEF).

The following is an example on how to use AOFEXDEF exit to assign a value to the CGLOBAL AOFRPCWAIT.

```
 aofrpcwait = '30'
'GLOBALV PUTC AOFRPCWAIT'
```

After modifying the exit, an SA OS/390 COLD START is required for these changes to take effect.

Table 5. Global Variables to Enable Advanced Automation (CGLOBALS)

Variable	Value	Effect
AOF_ASSIGN_JOBNAME	**1**	This indicates that SA OS/390 will exploit Tivoli NetView 1.2 ″ASSIGN BY JOBNAME″ feature. This is the default setting.
	0	SA OS/390 will not exploit Tivoli NetView 1.2 ″ASSIGN BY JOBNAME″ feature.
AOF_EMCS_AUTOTASK_ ASSIGNMENT	1	SA OS/390 will assign an autotask to extended MCS consoles with a console status of MASTER or ACTIVE
	0	SA OS/390 will not assign an autotask to extended MCS consoles with a console status of MASTER or ACTIVE 0 is the default.
AOF_EMCS_CN_ASSIGNMENT	**1**	SA OS/390 will obtain an extended MCS console with a unique name for operator station tasks (OSTs). If an MVS console was obtained for the OST previously, it will be released. 1 is the default setting.
	0	SA OS/390 will not obtain an extended MCS console with a unique name for OSTs and the command AOCGETCN will be disabled.
AOFACFINIT	**1**	This indicates that SA OS/390 will attempt to proceed with initialization despite error messages such as AOF722I during the processing of the automation control file. 1 is the default setting.
	0	SA OS/390 will stop the initialization process upon such errors.
AOFCTLOPT	YES	This indicates that SA OS/390 will ignore any IPL or RECYCLE options defined for a subsystem that has a status of CTLDOWN at SA OS/390 initialization.
	NO	SA OS/390 will honor all IPL and RECYCLE option definitions at SA OS/390 initialization. NO is the default setting.

Table 5. Global Variables to Enable Advanced Automation (CGLOBALS) (continued)

Variable	Value	Effect

AOFCNMASK		The characters that are used in determining unique console names can be tailored by updating the common global variable AOFCNMASK. This global is used as a hex mask to extract characters from the following string when generating unique console names with command AOCGETCN.
		`left(opid(),8)||right(opid(),8),`
		`||left(aofsysname,4)||right(aofsysname,4),`
		`||left(applid(),8)||right(applid(),8),`
		`||'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789$›#@_!?'`
		
		`Where: opid() is a function which returns the OST task name`
		`aofsysname is a common global which stores the system name`
		`applid() is a function which returns VTAM LU name`
		
		The default for AOFCNMASK is 290C0D0E0F101718. 29x selects character A in position 41, 0Cx - 10x selects the last five characters of the opid in positions 12 to 16, 17x and 18x select the last two characters of the sysname in positions 23 and 24. If AOFCNMASK is null, AOCGETCN will attempt to obtain a unique extended MCS console after a 1 minute interval, followed by a two minute interval and so forth for a maximum of 5 passes (15 minutes elapsed from the initial invocation of the command).
		
		For example:
		
		AOFCNMASK: 2A01020304051718
		
		2Ax selects character B in position 42, 01x - 05x selects the first five characters of the opid in positions 1 to 5, 17x and 18x select the last two characters of the sysname in positions 23 and 24.

AOFDEFAULT_TARGET	user defined	sets a default for the TARGET parameter for all commands where this parameter is used.
AOF_GATOPER_PW_CHANGE	**1**	The password for the Gateway operator will be changed every 30 days. 1 is the default setting.
	0	The password for the Gateway operator will not be changed.
AOFEXPLAIN_USER	user defined	The EXPLAIN command accepts this variable to include help support for customer installation supplied terms. It can hold one or more pairs of ***term/help panel*** specifications separated by a blank. If the specified satus in the EXPLAIN command is not a valid SA OS/390 status, the command routine will check whether it is an installation defined term. If so, the associated help panel is displayed.
AOFINITREPLY	**hh:mm**	The initial reply AOF603D is issued and automatically responded after hh:mm. 00:02 (2 minutes) is the default setting.
	0	The initial reply AOF603D will not be issued and automation continues with the default start without asking the operator.
AOF_INIT_MCSFLAG	user defined valid value	This variable contains the MCSFLAG that is used for WTOs and WTORs that are issued by SA OS/390 during initialization. The default is '00001000'.

Table 5. Global Variables to Enable Advanced Automation (CGLOBALS) (continued)

Variable	Value	Effect
AOF_INIT_ROUTCDE	user defined valid value	This variable contains the ROUTCDE (routing code) that is used for WTOs and WTORs that are issued by SA OS/390 during initialization. The default is '10000000'.
AOF_INIT_SYSCONID	user defined valid value	This variable contains the SYSCONID that is used for WTOs and WTORs that are issued by SA OS/390 during initialization. The default is '01'.
AOFLOCALHOLD	**0**	INGNTFY and SA OS/390 initialization will execute the SETHOLD AUTO command on the notify operator. 0 is the default setting.
	1	SETHOLD must be manually invoked.
AOFMINORCHK	**1**	Check automation settings for minor resources when automated commands are about to be issued in response to either a status change or a message, that is, in ACTIVMSG, HALTMSG, TERMMSG, ISSUECMD or ISSUEREP. This variable lets you use extended automation flags to govern the response to individual messages and status changes. It may also lead to some of your automation flag exits being rerun. Depending on how the exits are written this may or may not cause problems. More details are given in the description of the policy item MINOR RESOURCES in *System Automation for OS/390 Defining Automation Policy*. 1 is the default setting.
	0	SA OS/390 will not check extended automation flags before issuing the commands or replies.
AOFMOVOPT	YES	This indicates that SA OS/390 will ignore any IPL or RECYCLE options defined for a subsystem that has a status of MOVED at initialization. This does not apply to any subsystem that is defined to Automatic Restart Manager that SA OS/390 is aware of.
	NO	SA OS/390 will honor all IPL and RECYCLE option definitions at SA OS/390 initialization. NO is the default setting.
AOFPAUSE	**0** to 5	This is the number of seconds that SA OS/390 will allow for applications that have shut down to be cleared by MVS, in addition to their termination delay. As the AOFPAUSE value is applied to all applications it should be kept small. AOFPAUSE may be useful on a slow machine, where allowing an extra second or two before SA OS/390 checks if the application has been cleared could avoid the need to use a termination delay timer. No matter how AOFPAUSE is set, the application status will not be updated to AUTODOWN or CTLDOWN until SA OS/390 is sure that the application has been cleared from the system by MVS. 0 is the default setting.

Table 5. Global Variables to Enable Advanced Automation (CGLOBALS) (continued)

Variable	Value	Effect
AOFQUICKWTOR	**0**	Before a WTOR reply is responded to, an MVS display command is issued to check if it is outstanding. 0 is the default setting.
	1	Trust the CGLOBALS for WTOR reply processing.
AOFRELOADOPT	**No**	Will ignore the Start on RECYCLE option during an ACF reload. This is the default.
	Yes	Will honour the Start on RECYCLE option during an ACF reload. If Start on RECYCLE is NO then an inactive subsystem will be set to CTLDOWN.
AOFRESTARTALWAYS	**1**	An application that has been shut down normally, outside the control of SA OS/390, with RESTARTOPT=ALWAYS, will be restarted regardless of whether or not it has reached its critical error threshold. 1 is the default setting.
	0	An application that has been shut down normally, outside the control of SA OS/390, with RESTARTOPT=ALWAYS, will NOT be restarted if it has reached its critical error threshold.
AOFRPCWAIT	0 to n	This is the number of seconds that SA OS/390 will wait for command responses from other systems in the sysplex. **10** is the default setting.
AOFSERXINT	**1**	The exit AOFEXINT is processed under the BASEOPER automation operator under the initialization process. This is the default.
	0	The exit AOFEXINT execution is serialized within the initialization process.
AOF_SET_AVM_RESTART_EXIT	**1**	SA OS/390 will set the AVM restart exit during the initialization of the automation environment. **1** is the default.
	0	The AVM restart exit needs to be set in the SYS1.PARMLIB PROGxx member. Please refer to *System Automation for OS/390 Planning and Installation* for more information.
AOFSETSTATEOVERRIDE	**NO**	Sets the default OVERRIDE parameter for the SETSTATE command to NO. This is the default setting.
	FLG	Sets the default OVERRIDE parameter for the SETSTATE command to override all flag settings.
	TRG	Sets the default OVERRIDE parameter for the SETSTATE command to override all trigger settings.
	SVP	Sets the default OVERRIDE parameter for the SETSTATE command to override all service period settings.
	EVT	Sets the default OVERRIDE parameter for the SETSTATE command to override all event settings.
	ALL	Sets the default OVERRIDE parameter for the SETSTATE command to override all flag and trigger settings.

Table 5. Global Variables to Enable Advanced Automation (CGLOBALS) (continued)

Variable	Value	Effect
AOFSETSTATESCOPE	user defined valid value	allows you to override the predefined default for the SCOPE parameter of the SETSTATE command as documented in *System Automation for OS/390 Operator's Commands*.
AOFSETSTATESTART	user defined valid value	allows you to override the predefined default for the START parameter of the SETSTATE command as documented in *System Automation for OS/390 Operator's Commands*.
AOFSHUTDELAY	0 to 59	This is the number of minutes that SA OS/390 will wait for a termination message before continuing the shutdown process. Any values outside this range are treated as 0. With a setting of 0, message AOF745E will not be issued. **0** is the default setting.
AOFSHUTOVERRIDE	**NO**	Sets the default OVERRIDE parameter of the INGREQ command to NO. This is the default setting.
	FLG	Sets the default OVERRIDE parameter for the INGREQ command to override all flag settings.
	TRG	Sets the default OVERRIDE parameter for the INGREQ command to override all trigger settings.
	SVP	Sets the default OVERRIDE parameter for the INGREQ command to override all service period settings.
	EVT	Sets the default OVERRIDE parameter for the INGREQ command to override all event settings.
	ALL	Sets the default OVERRIDE parameter for the INGREQ command to override all flag and trigger settings.
AOFSPOOLFULLCMD	1	SA OS/390 will not execute the Spool recovery passes more than once. Message AOF2941I will be issued if the SPOOLFULL condition persists.
	0	SA OS/390 will re-execute the Spool recovery commands. 0 is the default setting.
AOFSPOOLSHORTCMD	1	SA OS/390 will not execute the Spool recovery passes more than once. Message AOF2941I will be issued if the SPOOLSHORT condition persists.
	0	SA OS/390 will re-execute the Spool recovery commands. 0 is the default setting.
AOFUSSWAIT	1 to n	This is the number of seconds SA OS/390 waits for the completion of a user-defined OS/390 UNIX monitoring routine (specified in the OS/390 UNIX Control Specification panel) until it gets a timeout. When the timeout occurs, SA OS/390 does no longer wait for a response from the monitoring routine and sends a SIGKILL to the monitoring routine. **10** is the default setting.
AOF3WTIME	1 to n	This is the number of seconds that the SHOWME command will wait for command responses. **10** is the default setting.
INGAUTO_INTERVAL	user defined valid value	sets the default for the INTERVAL parameter of the INGAUTO command.

Table 5. Global Variables to Enable Advanced Automation (CGLOBALS) (continued)

Variable	Value	Effect
INGREQ_ORIGINATOR	1	Indicates that SA OS/390 assigns individual originator IDs for each operator issuing an INGREQ command.
	0	All operators are grouped under originator ID OPERATOR. 0 is the default setting.
INGREQ_OVERRIDE	**NO**	Sets the default OVERRIDE parameter of the INGREQ command to the specified value. NO is the default setting.
	ALL TRG FLG DPY STS UOW INIT	
INGREQ_PRECHECK	NO	Sets the default PRECHECK parameter of the INGREQ command to the specified value.
	YES	YES is the default setting.
INGREQ_PRI	**LOW**	Sets the default priority (PRI parameter) of the INGREQ command to the specified value. LOW is the default setting.
	HIGH	
INGREQ_REMOVE	any allowed status	Sets the default value for the REMOVE parameter of the INGREQ command to the specified value. If the resource reaches the specified status (condition), the request is automatically removed.
INGREQ_RESTART	**NO**	Sets the default for the RESTART parameter of the INGREQ command when shutting down the resource. NO is the default setting.
	YES, CTL	
INGREQ_SCOPE	**ONLY**	Sets the SCOPE parameter of the INGREQ command to the specified value. ONLY is the default setting.
	ALL, CHILDREN	
INGREQ_TIMEOUT	**interval in minutes, MSG or CANCEL**	Sets the interval in minutes used to check for the INGREQ command used to check whether the request has been successfully completed, and whether to send a message or cancel the request if it has not been satisfied after that time.
INGREQ_TYPE	**NORM**	Sets the default startup/shutdown type (TYPE parameter) of the INGREQ command to the specified value. NORM is the default setting.
	IMMED, FORCE	

Table 5. Global Variables to Enable Advanced Automation (CGLOBALS) (continued)

Variable	Value	Effect
INGREQ_VERIFY	**YES**	Sets the VERIFY parameter of the INGREQ command to YES. YES is the default setting for operator tasks.
	NO is the default setting for unattended tasks.	
INGSET_VERIFY	**YES**	Sets the default VERIFY parameter of the INGSET command to YES. YES is the default setting for operator tasks.
	NO is the default setting for unattended tasks.	
INGVOTE_VERIFY	**YES**	Sets the default VERIFY parameter of the INGVOTE command to the specified value. YES is the default setting for autotasks.
	NO is the default setting for operator tasks.	

Appendix C. Customizing the Status Display Facility (SDF)

Overview of Status Display Facility

This appendix explains how to customize SDF panels, descriptors, and operations.

How SDF Works

The SA OS/390 Status Display Facility (SDF) uses colors and highlighting to represent subsystem resource states. Typically, a subsystem shown in green on the SDF status panel indicates it is up, while red indicates a subsystem in a stopped or problem state. SDF can be tailored to present the status of system components in a hierarchical manner.

Note: SDF works only with MVS systems and resources.

Types of SDF Panels

Figure 51 on page 158 shows several SDF screens for system CHI01. This figure shows the main types of panels used in SDF.

- The *root component*
- The *status component*
- The *detail status display*

In addition to these panel types, you can create other types of panels according to your system requirements and the applications you are monitoring.

Note: All SDF panels must contain 24 rows and 80 columns. Because SDF uses only the display's default screen size, the default size must be defined as 24 x 80.

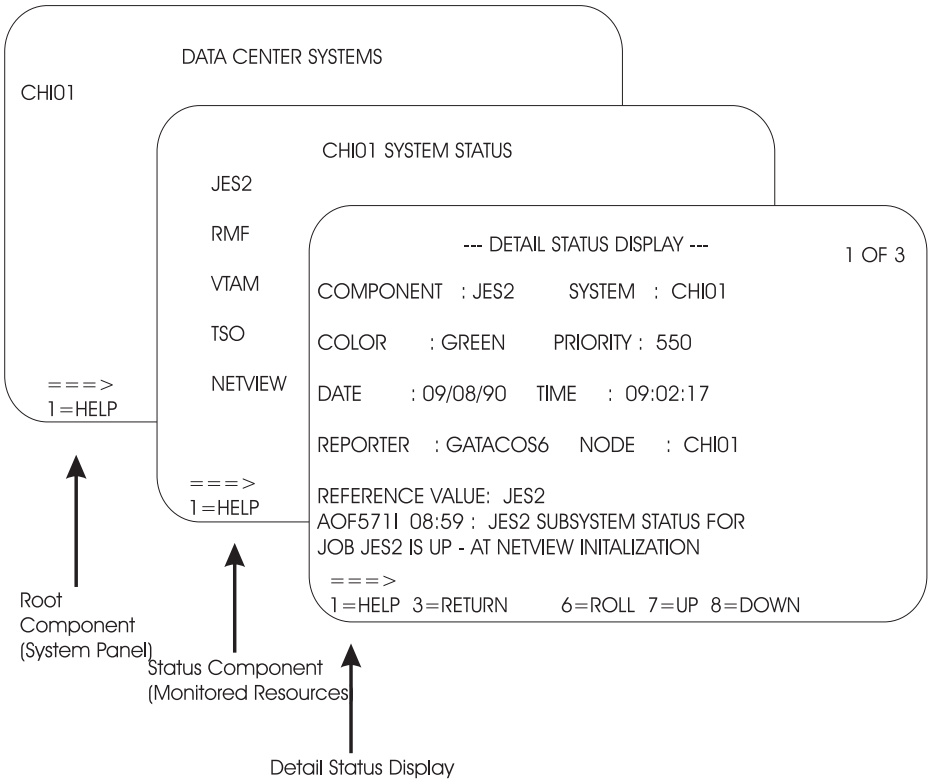


Figure 51. Example SDF Panels

Root Component
The root component is typically an element appearing on the first screen displayed when SDF is started. In Figure 51, the CHI01 system is the root component.

Status Component
Resources monitored by SDF are called *status components*. In Figure 51, system CHI01 has JES2, RMF, VTAM, TSO, and NetView status components, as shown on the CHI01 System Status panel. The status component panel displays all monitored resources in a system. Each monitored resource is shown in the color of its current status. For example, JES2 is shown in green if it is up.

Detail Status Display
A detail status display is built from information in a status descriptor (see "Status Descriptors"). This panel is displayed by tabbing to the appropriate resource on the status component panel and pressing the detail PF key. Each status component can have one or more status descriptors, or detail records, associated with it.

Figure 51 shows an example detail status display for a JES2 status descriptor. The `1 of 3` on the panel indicates that JES2 currently has three status descriptors, and therefore three detail status displays, associated with it.

Status Descriptors
A *status descriptor* is a detailed record of information about a resource status. In its raw form, a status descriptor is a multiline SA OS/390 message containing information such as:

- Root component and status component to which the status descriptor applies

- Priority, color, and highlighting associated with the status descriptor (see "How Status Descriptors Affect SDF" on page 160 for more information)
- Date and time the status descriptor was generated
- Actual resource status information; for example, an SA OS/390 message indicating the resource is up

SDF uses information in a status descriptor to generate a detail status display (see "Detail Status Display" on page 158). You do not usually look directly at a status descriptor; rather, you look at portions of it through a detail status display. For example, in Figure 51 on page 158, the detail status display presents information from a status descriptor for status component JES2. The `1 of 3` on the panel indicates that JES2 currently has three status descriptors associated with it.

SDF generates, displays, and deletes status descriptors.

SDF Tree Structures

SDF uses *tree structures* to set up the hierarchy of monitored resources displayed on SDF status panels. An SDF tree structure always starts with the system name as the root node and has a level number of one. Tree structure levels subordinate to the root node are the monitored resources. The level numbers of these resources reflect their dependency on each other.

You define SDF tree structures in NetView DSIPARM data set member AOFTREE.

Figure 52 on page 160 shows an example SDF tree structure. Following the tree structure definition statements is a diagram showing how these statements result in a tree structure.

```
1 SY1
  2 SYSTEM
    3 WTOR
    3 APPLIC
      4 AOFAPPL
        5 AOFSSI
      4 JES
      4 VTAM
    3 TSO
    3 RMF
  2 GATEWAY
```

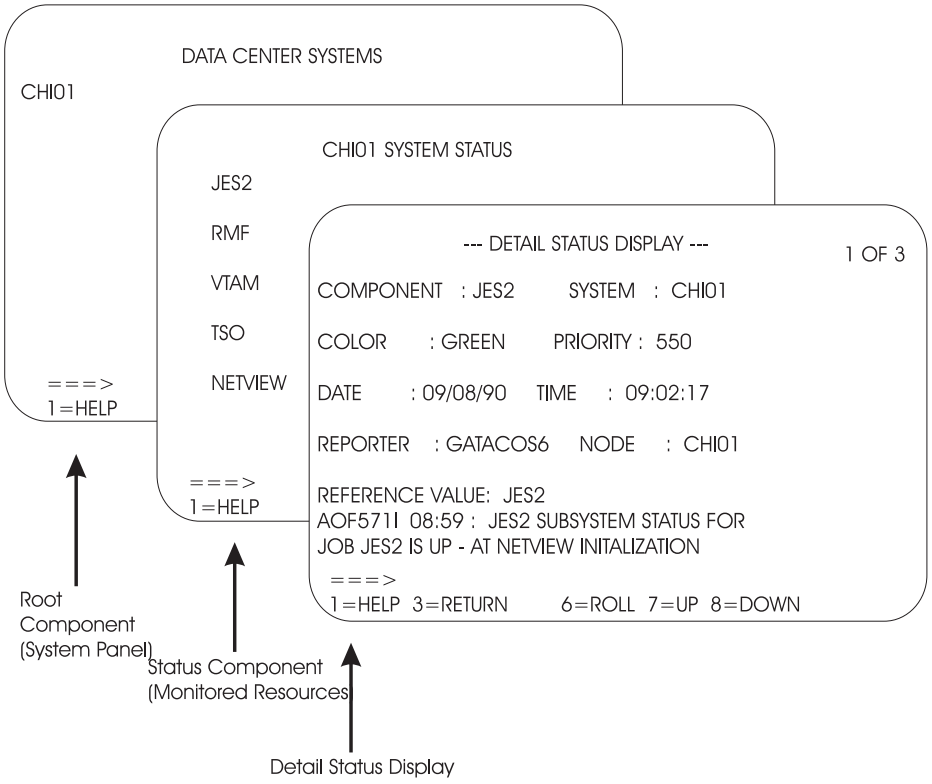


Figure 52. Example SDF Tree Structure

SA OS/390 supplies a sample SDF tree structure in the SA OS/390 sample library. This tree structure is referenced by a %INCLUDE statement in member AOFTREE in the NetView DSIPARM data set. You can customize this sample tree structure to meet your requirements. This order of dependency does *not* have to be the same as that used for system startup or shutdown using SA OS/390.

For example, using the tree structure in Figure 52, if there is a problem with TSO, it is not desirable to also change the VTAM status color, because VTAM is not having any problems. In contrast, in the SA OS/390 startup and shutdown procedures, TSO is dependent on VTAM.

More details on SDF tree structure definitions are in "Step 1: Defining SDF Hierarchy" on page 168.

How Status Descriptors Affect SDF

Status descriptors are the main units of information SDF uses. The information in status descriptors determines how your SDF status displays look at any point in time. This section explains how SDF uses status descriptors.

Priority and Color Assignments

Status descriptors are assigned both a priority number and a color. These color and priority assignments determine the colors in which status components are displayed. In SDF, a lower number indicates a higher priority. Status descriptors are connected to the status component in ascending order of priority.

Color and priority assignments for status descriptors are defined in two places:

- In the PRIORITY parameter in the AOFINIT member of the NetView DSIPARM data set. This parameter defines initial priority and color assignments used for status descriptors. The values defined in AOFINIT are used if no further customization is done to priority and color assignments. The default priority ranges and colors used in AOFINIT are:

Priority Range	Color
001 to 199	Red
200 to 299	Pink
300 to 399	Yellow
400 to 499	Turquoise
500 to 599	Green
600 to 699	Blue

  White is used as the default status descriptor color (the DCOLOR parameter in member AOFINIT, described in *System Automation for OS/390 Programmer's Reference*) and as the default color for a status component without a tree structure entry (the ERRCOLOR parameter in member AOFINIT, described in *System Automation for OS/390 Programmer's Reference*). For more information on the PRIORITY parameter, see *System Automation for OS/390 Programmer's Reference*.
- In the SDF definitions in the Status Details policy object. These entries define colors, highlighting, and priorities used for particular resource statuses. Color and priority assignments defined in the customization dialog can be used to override assignments in the AOFINIT member.

  Note: Some of the resource statuses that appear in SDF displays do not directly correspond to resource statuses used in the automation status file.
  System Automation for OS/390 User's Guide shows the default resource status types, colors, highlighting, and priorities provided with SA OS/390. These settings define to SA OS/390 the parameters used when adding status descriptors to SDF.

  For more information on the SDF Status Details definition, see "Step 4: Defining SDF in the Customization Dialog" on page 172.

Chaining of Status Descriptors to Status Components

A resource status change causes a status descriptor to be generated. SDF adds this status descriptor to a chain of status descriptors. Chained status descriptors determine the status and color of status components. The highest-priority status descriptor in a chain determines the initial color in which the status component is displayed. The underlying chained priority numbers determine the color in which successive detail status displays will be shown.

Status descriptors are chained off each level of status component in a tree structure. Status descriptors chained to lower-level status components are also

chained to a higher-level status component, again in order of priority. Status descriptors are also chained off the root component. These status descriptors are all the status descriptors that currently exist at all levels of the tree structure.

For example, Figure 53 shows status descriptors currently generated for system SY1. The priority for each status descriptor is shown by a number.

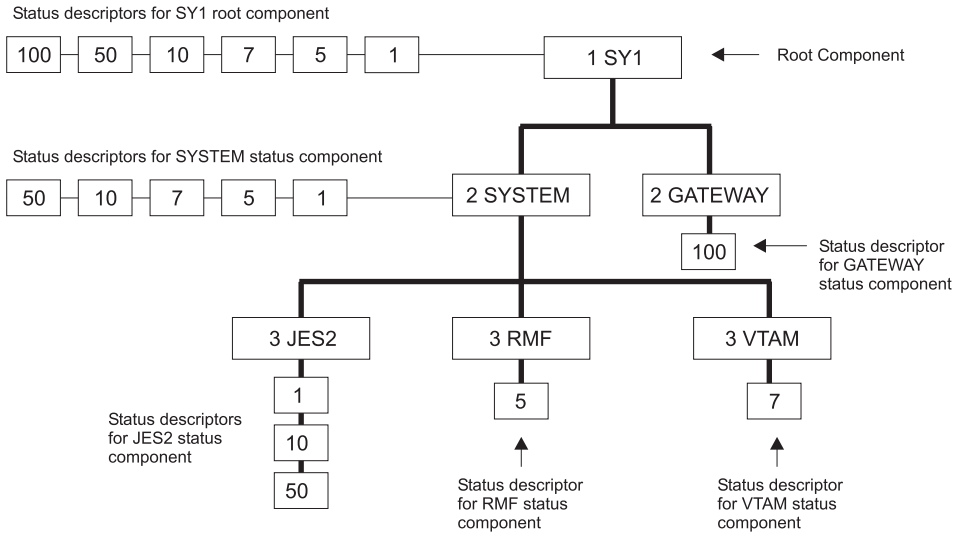


Figure 53. Status Descriptors Chained to Status Components

The status components at the lowest level in this tree structure, JES2, RMF, and VTAM, have status descriptors chained off them. Status component JES2 has three status descriptors chained, with priorities 1, 10, and 50. Because 1 is the highest priority, the status descriptor with priority 1 is organized first in the chain. This highest-priority status descriptor determines the color in which JES2 is displayed on the status panel. If an operator uses the detail PF key to view detail status displays for JES2, the information contained in the status descriptor with priority 1 will be displayed first, then the detail status display for the status descriptor with priority 10, and so on.

At the SYSTEM status component level in the tree structure, all status descriptors from the lower-level status components are also chained. Because the status descriptors chained to RMF and VTAM have higher priorities than the priority 10 and 50 status descriptors for JES2, they are organized after the priority 1 status descriptor in the chain. An operator using the detail PF key at the SYSTEM level could view five detail status displays, ranging from priority 1 to priority 50.

Similarly, at the SY1 level in the tree structure, all status descriptors chained to all status components in the tree structure are chained in order of priority. An operator using the detail PF key at the SY1 level could view six detail status displays, ranging from priority 1 to priority 100.

If a status component has multiple status descriptors with equal priorities, the status descriptors are chained off the status component in order of arrival time.

When a status descriptor no longer accurately reflects the actual status of a resource, SDF automatically deletes it from status descriptor chains. As an example of how priority determines order of status descriptors, suppose two status descriptors currently exist for status component JES2. If there are two status

descriptors for JES2 with priorities of 120 and 140, the status descriptor with priority 120 is displayed first. In both cases, JES displays in red on the SDF status panel.

In SA OS/390, all status types are defined in the automation control file. When an automation event occurs, the SA OS/390 AOCUPDT common routine scans the automation control file for the SDF entry for that status. SA OS/390 issues a request to add the status using the information from the automation control file.

For example, suppose subsystem RMF, shown on the example SDF panels in Figure 51 on page 158, is set to a STOPPING state. The SA OS/390 AOCUPDT common routine scans the automation control file for the STOPPING state entry for SDF and generates a status descriptor, specifying a priority of 330. SDF adds the status descriptor to the RMF status component. RMF appears as yellow and blinking on the status panel. Once RMF is in a stopped state, the AOCUPDT common routine scans the automation control file for the STOPPED state SDF entry and generates a status descriptor with priority 130. SDF adds this new status descriptor to the RMF status component. Now, RMF appears in red on the SDF status panel.

Propagating Status Descriptors Upward and Downward in a Tree Structure

Based on the order of dependencies defined in a tree structure, status descriptors can be *propagated* upward or downward to status components in a tree structure. This propagation of status descriptors affects the color in which status components are displayed, as well as the detail status displays operators can view by using the detail PF key on a particular status component.

Propagation of status upward and downward in a tree structure is defined by the PROPUP and PROPDOWN parameter in the AOFINIT member (see *System Automation for OS/390 Programmer's Reference* for descriptions).

The SA OS/390-provided defaults for status propagation in the AOFINIT member are to propagate status upward (PROPUP=YES) but not downward (PROPDOWN=NO).

When status is propagated upward in a tree structure, if a status descriptor is added or deleted at a lower level in the tree structure, it is also added or deleted from the cumulative chain of status descriptors at a higher-level node in the tree structure.

Propagation of status upward in a tree structure consolidates the status of all monitored resources in the system at the root node. In this way, the color of the root node reflects the most important or critical status in a computer operations center. For example, in Figure 52 on page 160, any color changes for AOFSSI are reflected in AOFAPPL, APPLIC, SYSTEM, and SY1, if SDF propagates status changes upward in the tree structure. In Figure 51 on page 158, if all monitored resources are green, the root node CHI01 on the Data Center Systems panel is also shown in green.

When status is propagated downward in a tree structure, if a status change occurs at a higher level in a tree structure, the changes are sent downward in the tree structure. This propagating downward could cause status descriptors at lower levels in the tree structure to be added or deleted.

Propagating status downward can be useful when an entire system is down. In such a case, you want SDF status panels to accurately reflect the system status. You do not want status components lower in the tree structure to retain previously generated status descriptors indicating that the components are up and running, because these status descriptors do not accurately reflect the status of the components. You can configure your SDF implementation to propagate status downward, and remove all status descriptors from all status components in a tree structure. If an operator tries displaying detailed status about any of the status components lower in the tree structure, they receive "NO DETAIL INFO AVAILABLE" messages. The empty chain color, defined by the EMPTYCOLOR parameter in member AOFINIT with a default color of blue, is also used to indicate that no detail information is available. See *System Automation for OS/390 Programmer's Reference* for the EMPTYCOLOR description.

How SDF Helps Operations to Focus on Specific Problems

SDF structure and processing allows the program identifying a problem to be concerned only with the specific problem.

For example, suppose an application program detects a warning message for status component JES on CHI01. The following processing steps occur:

1. The application program issues a request to SDF to add a status descriptor for JES.
2. The status entry for JES on system CHI01 now indicates there is a problem with JES. If the SDF is configured to propagate status up the hierarchical tree structure, the status for system CHI01 also reflects the problem state. See *System Automation for OS/390 Programmer's Reference* for details on the PROPUP SDF initialization parameter.
3. Now, suppose another more serious problem occurs. The application program which detects this new problem issues another request to SDF to add a status descriptor having a lower priority number than the status descriptor for the first problem.
4. Because status descriptors are chained in order of priority, the JES status now reflects the status descriptor color of the more serious problem.
5. When the more serious problem is resolved, the application program detecting the problem resolution issues a request to SDF to remove the status descriptor for this problem from the chain of JES status descriptors.
6. The status panel is updated to reflect the first problem.

How SDF Panels Are Defined

All SDF status panels, apart from detail status display panels, are defined in the AOFPNLS member of the NetView DSIPARM data set.

Member AOFPNLS can contain either one or both of the following:

- %INCLUDE statements referencing other NetView DSIPARM members containing definitions of panels. The %INCLUDE statement causes the named panel definition member to be loaded. This is the recommended method, and the method used in the SA OS/390-provided version of AOFPNLS.
- Panel structure definitions for all SDF panels.

Panel members defined or referenced in AOFPNLS are loaded into system memory, and may be deleted, replaced, or temporarily made resident using the SDFPANEL command (see *System Automation for OS/390 Programmer's Reference* for command description).

Panels that are to be dynamically loaded as needed (see "Dynamically Loading Tree Structure and Panel Definition Members") must be defined in a NetView DSIPARM member having the same member name as the panel itself.

It is recommended that you include only frequently used panels in AOFPNLS, to conserve system memory. Other panels can be dynamically loaded when needed, either by pressing a SDF function key or by using the SCREEN command.

Note: Dynamic refresh will only work with panels defined in AOFPNLS.

SDF internally formats and builds detail status display panels from the information in a status descriptor. You do not have to define and format detail status display panels. Status components defined in the panel definitions must also be defined in the corresponding tree structure. However, not all status components defined in the tree structure require a corresponding entry on the SDF status panel. For example, in Figure 52 on page 160, the APPLIC status component is only a pseudo-entry and may not actually be displayed on any SDF status display panel.

SDF status panels can be customized to reflect any environment. For example, you can define a panel to show the status of all JES subsystems on all processors in a computer operations center. The JES operator can view the panel to determine the status of any JES subsystem in the complex.

For detailed information on defining SDF panels, see "Step 2: Defining SDF Panels" on page 169.

Dynamically Loading Tree Structure and Panel Definition Members

Using %INCLUDE statements in the main SDF tree structure and panel definition members allows you to dynamically load tree structure and panel definition members without restarting SDF (see *System Automation for OS/390 Programmer's Reference*). The SDFTREE command loads a tree structure definition member. The SDFPANEL command loads a panel definition member. You can dynamically reload members AOFTREE and AOFPNLS themselves.

Using SDF for Multiple Systems

You can configure SDF so that multiple systems in an automation network can forward their resource status information to the SDF on the focal point system. In a multiple-system environment, the following must be defined:

- The tree structure for each system must be defined in the AOFTREE member of NetView DSIPARM on the focal point system SDF. The root name must be unique for each system tree structure.
- The focal point root name must match the SYSNAME value defined in the automation policy. This value is specified in the customization dialog.

  Note: The SYSNAME for each system under SA OS/390 control must be the same as the system name under which the system was IPLed
- For target system SDF status update to occur on a focal point SDF, SA OS/390 focal point services must already be implemented.

Because each root name must be unique in a multiple-system environment, any status component on any system defined to the focal point SDF can be uniquely addressed by prefixing the status component with the root component name:

```
ROOT_COMPONENT.STATUS_COMPONENT
```

For example:

```
SY1.JES2
```

Similarly, any SDF status descriptors forwarded from the target system to the focal point SDF are prefixed with the root name of the target system by SA OS/390 routines.

SDF Components

SDF consists of the following components:

Table 6. SDF Components

Name	Type	Purpose
AOFTDDF	Task	Initializes SDF and maintains the status database. This initialization is an automated function.
SDF	Command	Starts an SDF operator session.
SDFTREE	Command	Dynamically loads or deletes an SDF tree structure definition member from the NetView DSIPARM data set.
SDFPANEL	Command	Dynamically loads or deletes an SDF panel definition member from the NetView DSIPARM data set.
AOFINIT	Input file	Contains SDF initialization parameters defined with the statements described in *System Automation for OS/390 Programmer's Reference*. AOFINIT is in the NetView DSIPARM data set.
AOFTREE	Input file	Contains tree structures described in *System Automation for OS/390 Programmer's Reference*. This member usually consists of a list of %INCLUDE statements referencing other members containing tree structures. AOFTREE is in the NetView DSIPARM data set.
AOFPNLS	Input file	Contains SDF panel parameters defined by the statements described in "Step 2: Defining SDF Panels" on page 169. This member usually consists of a list of %INCLUDE statements referencing other members containing panel definitions. AOFPNLS is in the NetView DSIPARM data set.
panel_name	Input file	A DSIPARM member containing the definition of one or more SDF panels or %INCLUDE statements identifying other DSIPARM panel definition members. It is highly recommended that panel definition members contain the definition of a single panel having the same name as the member.
tree_name	Input file	A DSIPARM member containing the definition of one or more tree structures. It is highly recommended that tree definition members contain the definition of a single tree having the same root component name as the member name.

How the SDF Task Is Started and Stopped

During SA OS/390 initialization, the AOFTDDF task loads members defining panel format, panel flow, and tree structures. Member AOFINIT defines parameters common to all SDF panels and basic initialization specifications, such as screen size, default PF keys, and the initial screen displayed when a SDF session is started. These AOFINIT parameters are described in *System Automation for OS/390 Programmer's Reference*.

Starting the SDF Task

In SA OS/390 code, the AOFTDDF task is started by the following command:

```
START TASK=AOFTDDF
```

Stopping the SDF Task

In SA OS/390 code, the AOFTDDF task is stopped by the following command:

```
STOP TASK=AOFTDDF
```

Note: When SDF is restarted, all existing SDF status descriptors are lost, as they are kept only in memory.

SDF Definition

The following section describes the SDF definition process.

Summary of SDF Definition Process

This section summarizes the steps for defining the SDF. Use this procedure to define the panels displayed in an SDF session. Details on each step are provided later in this chapter and in *System Automation for OS/390 Programmer's Reference*.

1. Define the hierarchy of monitored resources used for your SDF panels, using tree structure statements in NetView DSIPARM data set members. These tree structure definition members should be referenced by %INCLUDE statements in the main SDF tree structure definition member, AOFTREE, in the NetView DSIPARM data set. See *System Automation for OS/390 Programmer's Reference* for details.
2. Define SDF status panels using panel definition statements in NetView DSIPARM data set members. Panels can either be automatically loaded when SDF starts, or dynamically loaded using the SDFPANEL command. For panels to be automatically loaded, add a %INCLUDE statement specifying the panel definition member to the main panel definition member, AOFPNLS, in the NetView DSIPARM data set. See "Step 2: Defining SDF Panels" on page 169 for details.

   Define and customize SDF status panels in the following general order:
   a. Root panel
   b. Status component panel for each entry on the root panel
   c. Any other customized status panels.
3. Customize the SDF initialization parameters in NetView DSIPARM member AOFINIT, if necessary (optional), or use defaults. See *System Automation for OS/390 Programmer's Reference* for detailed descriptions of SDF initialization parameters. Using defaults is recommended.
4. Define SDF resource status, color, highlight and priority values using the customization dialog to edit the SDF Status Details policy object, or use defaults. This step is optional. See *System Automation for OS/390 Defining Automation Policy* for the description of the Status Details policy object. Using defaults is recommended.

Notes:

1. Resources that SA OS/390 is not currently automating are not displayed on SDF panels.
2. To display the status of multiple systems and forward status from target systems to SDF on a focal point system, SA OS/390 focal point services must already be implemented. See *System Automation for OS/390 Defining Automation Policy* for details on configuring focal point services.

Step 1: Defining SDF Hierarchy

Member AOFTREE in the NetView DSIPARM data set contains a set of definitions that define the propagation hierarchy for status color changes. When the status changes for a component, the corresponding color change is propagated up or down the tree to the next higher or lower level component. The level is determined by the level number assigned to each component. The type of propagation is determined either by the entry in the AOFINIT member or by individual requests to add a status descriptor to a status component.

Note: SA OS/390 does not use this SDF hierarchy for subsystem shutdown or startup procedures. Instead, SA OS/390 uses subsystem entries defined in the automation policy to determine startup and shutdown relationships and hierarchies.

Tree Structure Definitions

AOFTREE contains tree structure definitions. To define tree structures, you can:

- Use %INCLUDE statements referencing other members containing definitions for specific tree structures. This is the recommended method, and the method used in the SA OS/390-provided version of AOFTREE.

  On the %INCLUDE statement, the name of the referenced member must be enclosed in parentheses.
- Place all tree structure definitions in AOFTREE.
- Use a combination of both.

Figure 54 shows a typical tree structure definition:

```
1 SY1
  2 APPLIC
     3 AOFAPPL
       4 AOFSSI
     3 JES
     3 VTAM
     3 TSO
     3 RMF
  2 GATEWAY
```

Figure 54. Example Tree Structure Definition

In this tree structure, SY1 is the root component. This definition is in a separate member, named SY1. It is referenced by the following statement in the AOFTREE member:

```
%INCLUDE(SY1TREE)
```

Loading Tree Structures: All tree structures need not be loaded during initialization. Some can be loaded dynamically after SDF is started. To do this, use AOFTREE to define those tree structure entries that will be loaded during initialization, then, use the SDFTREE command to load additional tree structures as needed. For more information, see *System Automation for OS/390 Programmer's Reference*.

Tree structures loaded after SDF is started must be contained in separate members. Each member must be named after the root component for which the tree structure is defined.

Step 2: Defining SDF Panels

SDF status panels are defined in NetView DSIPARM member AOFPNLS. SA OS/390 loads the panel definitions in AOFPNLS when SDF is initialized.

Panel Definition Methods

To define panels in AOFPNLS, you can:

- Use %INCLUDE statements referencing separate NetView DSIPARM members containing panel definitions. This is the recommended method, and the method used in the SA OS/390-provided version of AOFPNLS. See "%INCLUDE Statement for SDF Panels" on page 171 for details on using the %INCLUDE statement for SDF panel definition members.
- Include actual definitions for all panels.
- Use a combination of both %INCLUDE statements and panel definitions.
- Include a subset of panel entries to load during initialization, so that additional panel definitions can be loaded only when needed (see *System Automation for OS/390 Programmer's Reference*).

Panel Definition Structure

The structure of each panel definition is as follows:

- Begin panel definition statement (PANEL)
- Status component definition statements, consisting of pairs of the following statements:
  - STATUSFIELD: defines location of a status component on a panel
  - STATUSTEXT: defines the text displayed in the STATUSFIELD
- Text fields and data definition statements, consisting of pairs of the following statements:
  - TEXTFIELD: defines locations and attributes for constant fields on panels
  - TEXTTEXT: defines text displayed in the TEXTFIELD
- Status panel PF key definitions (PFKnn)
- End panel statement (ENDPANEL)

Descriptions of these panel definition statements are in *System Automation for OS/390 Programmer's Reference*.

Recommended Order for Defining Panels

When defining panels, it is recommended that you define them in the following order:

1. The root panel
2. The status components for each item listed on the root panel
3. Any other customized status panels

Note: This order of defining panels is a recommendation only. You can define your SDF panels in any order desired.

Example Panel Definition

Figure 55 on page 170 shows how an SDF panel looks when displayed:

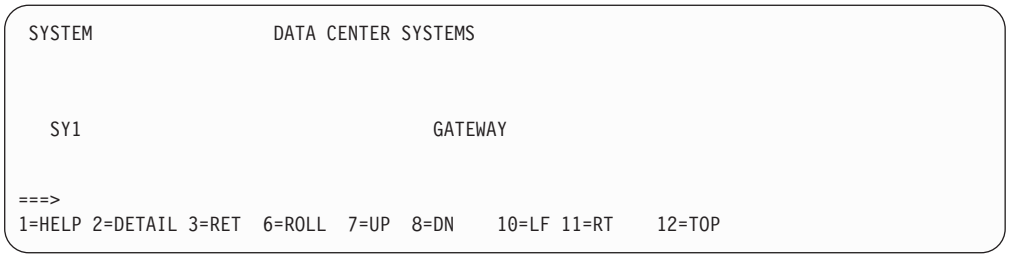


Figure 55. Example SDF Panel

Figure 56 shows the panel definition statements required to define the panel in Figure 55.

```
PANEL(SYSTEM,24,80)
TEXTFIELD(01,02,10,WHITE,NORMAL)
TEXTTEXT(SYSTEM)
TF(01,25,57,WHITE,NORMAL)
TT(DATA CENTER SYSTEMS)
STATUSFIELD(SY1,04,04,11,N,,SY1SYS)
STATUSTEXT(SY1)
SF(SY1.GATEWAY,02,40,47,N,,GATEWAY)
ST(GATEWAY)
TF(24,01,79,T,NORMAL)
TT(1=HELP 2=DETAIL 3=RET  6=ROLL  7=UP 8=DN    ,
10=LF 11=RT 12=TOP)
PFK1(AOCHELP SDF)
PFK2(DETAIL)
PFK3(RETURN)
PFK6(ROLL)
PFK7(UP)
PFK8(DOWN)
PFK10(LEFT)
PFK11(RIGHT)
PFK12(TOP)
ENDPANEL
```

Figure 56. Example Panel Definition Entry

In Figure 56, the panel name is SYSTEM. This panel definition can either be in a separate member referenced by a %INCLUDE statement in AOFPNLS or be directly coded in AOFPNLS. The recommended method is to use a separate member and a %INCLUDE statement. If it is in a separate member, the member name is SYSTEM. You do not have to explicitly define every PF key for the panel. PF key definitions not specified are picked up from definitions in NetView DSIPARM member AOFINIT.

Table 7 describes each statement in Figure 56:

Table 7. Panel Definition Entry Description

Statement	Description and Example Value
`PANEL(SYSTEM,24,80)`	The panel definition statement. The panel name is SYSTEM, the panel length is 24, and the panel width is 80.
`TEXTFIELD(01,02,10,WHITE,NORMAL)`	The text location statement defining constant panel fields. This field starts on line 01 in position 02 and ends in position 10. The color of the field is white and highlighting is normal.

Table 7. Panel Definition Entry Description (continued)

Statement	Description and Example Value
`TEXTTEXT(SYSTEM)`	The text data statement specifying the actual data that goes in the text field just defined. This field contains the word SYSTEM. TEXTFIELD and TEXTTEXT are always grouped in pairs.
`TF(01,25,57,WHITE,NORMAL)`	Another TEXTFIELD statement for another constant field.
`TT(DATA CENTER SYSTEMS)`	Another TEXTTEXT statement for the text field just defined.
`STATUSFIELD(SY1,04,04,11,N,,SY1SYS)`	The location of the status component field. The status component is *SY1*. This field starts on line 04 in position 04 and ends in position 11. The highlighting level is normal. The next panel displayed when the Up PF key is pressed is SY1SYS.
`STATUSTEXT(SY1)`	The text data used for the name of the field just defined with the STATUSFIELD statement. In this case, the field name is SY1. STATUSFIELD and STATUSTEXT statements are grouped in pairs.
`SF(SY1.GATEWAY,02,40,47,N,,GATEWAY)`	Another STATUSFIELD definition.
`ST(GATEWAY)`	Another STATUSTEXT definition.

`TF(24,01,79,T,NORMAL)`
`TT(1=HELP 2=DETAIL 3=RET 6=ROLL 7=UP,`
`8=DN 10=LF 11=RT 12=TOP)`
	Here, TEXTFIELD and TEXTTEXT are used to display PF key definitions. For this panel, these are the default definitions defined in AOFINIT. If you need values differing from the defaults, there is a statement for defining PF keys unique to this panel, DPFKnn. See *System Automation for OS/390 Programmer's Reference* for a description of this statement.

`PFK1(AOCHELP SDF)`
`PFK2(DETAIL)`
`PFK3(RETURN)`
`PFK6(ROLL)`
`PFK7(UP)`
`PFK8(DOWN)`
`PFK10(LEFT)`
`PFK11(RIGHT)`
`PFK12(TOP)`
	PF key definition statements.

`ENDPANEL`	The end panel statement, indicating that this is the end of definitions for this panel.

%INCLUDE Statement for SDF Panels

The %INCLUDE statement for SDF has the following features:

- The SDF %INCLUDE statement allows specifying a list of members rather than a single member only. Each member name in the list represents a DSIPARM member be loaded. Member names in the list are delimited by a comma.
- The SDF %INCLUDE statement requires parentheses around the specified member or members.
- The target DSIPARM members may contain only complete panel definitions or additional %INCLUDE statements. Panel definitions must be contained within a single member, and therefore cannot be built using commonly defined segments.

Step 3: Customizing SDF Initialization Parameters

Member AOFINIT allows you to define parameters common to all SDF panels and SDF initialization specifications, such as:

- Initial screen shown when SDF is started
- Maximum operator logon limit

- Default PF key definitions
- Detail status display panel PF key definitions
- Detail status display panel PF key descriptions
- Default priorities and colors

These parameters define values for SDF when it is started.

This step of SDF customization is optional. Using SA OS/390-provided default values for these parameters is recommended.

Note: User-defined statuses are not saved across a recycle or a monitor cycle. This means the status of a subsystem will change from the user-defined status to an appropriate SA OS/390 status.

Step 4: Defining SDF in the Customization Dialog

The SDF entries in the Status Details policy object allow you to define statuses and the priorities assigned to those statuses. These entries are used by SA OS/390 common routines to gather data for requests to add status descriptors to status components. The format and values used in SDF Status Detail definitions are described in *System Automation for OS/390 Programmer's Reference*.

This step of SDF customization is optional. Using SA OS/390-provided definitions for SDF is recommended.

Glossary

This glossary includes terms and definitions from:

- The *IBM Dictionary of Computing* New York: McGraw-Hill, 1994.
- The *American National Standard Dictionary for Information Systems* , ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies can be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018. Definitions are identified by the symbol (A) after the definition.
- The *Information Technology Vocabulary* developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.

The following cross-references are used in this glossary:

Contrast with. This refers to a term that has an opposed or substantively different meaning.

Deprecated term for. This indicates that the term should not be used. It refers to a preferred term, which is defined in its proper place in the glossary.

See. This refers the reader to multiple-word terms in which this term appears.

See also. This refers the reader to terms that have a related, but not synonymous, meaning.

Synonym for. This indicates that the term has the same meaning as a preferred term, which is defined in the glossary.

Synonymous with. This is a backward reference from a defined term to all other terms that have the same meaning.

A

ACF. Automation control file.

ACF/NCP. Advanced Communications Function for the Network Control Program. See *Advanced Communications Function* and *Network Control Program*.

ACF/VTAM*. Advanced Communications Function for the Virtual Telecommunications Access Method. Synonym for *VTAM*. See *Advanced Communications Function* and *Virtual Telecommunications Access Method*.

ACO. Automated console operations.

active monitoring. In SA OS/390, 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 OS/390, the system defined as the communication path between two SA OS/390 systems that do not have a direct link. An adjacent NetView is used for message forwarding and as a communication link between two SA OS/390 systems. For example, the adjacent NetView is used when sending responses from a focal point system 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 the NetView program, a high-priority event that warrants immediate attention. A database record is generated for certain event types that are defined by user-constructed filters.

alert condition. A problem or impending problem for which some or all of the process of problem determination, diagnosis, and resolution is expected to require action at a control point.

alert focal point system. See entry for NPDA focal point system under *focal point system*

alert threshold. An application service value that determines the level at which SA OS/390 changes the associated icon on the NMC workstation to the alert color. SA OS/390 may also issue an alert. See *warning threshold*

AMC. (1) Automation Manager Configuration (2) The Auto Msg Classes entry type

APF. Authorized program facility.

APPC. Advanced program-to-program communications.

application. An OS/390 subsystem or job monitored by SA OS/390.

Application entry. A construct, created with the customization dialog, used to represent and contain policy for an application.

application group. A named set of applications. An application group is used to represent and contain common policy for the contained applications. Applications in an application group can be automated as an entity. Application groups can behave differently according to their defined nature (BASIC, MOVE, SERVER). An application group can also be used for monitoring purposes.

ApplicationGroup entry type. A construct, created with the customization dialog, used to represent and contain policy for an application group.

application program. (1) A program written for or by a user that applies to the user's work, such as a program that does inventory or payroll. (2) A program used to connect and communicate with stations in a network, enabling users to perform application-oriented activities.

ARM. Automatic restart management.

ASCB. Address space control block.

ASCB status. An application status derived by SA OS/390 running a routine (the ASCB checker) which searches the OS/390 address space control blocks (ASCBs) for address spaces with a particular job name. The job name used by the ASCB checker is the job name defined in the customization dialog for the application.

ASCII (American National Standard Code for Information Interchange). The standard code, using a coded character set consisting of 7-bit coded characters (8-bits including parity check), for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters. (A)

ASF. automation status file.

assist mode facility. An SA OS/390 facility that uses SDF and enables interaction with automation before SA OS/390 takes an automation action. SDF prompts the operator with a suggested action, then provides options for using that action, modifying and using the action, or canceling the action. Also called assist mode, it is enabled using the customization dialog, or dynamically.

authorized program facility (APF). A facility that permits identification of programs that are authorized to use restricted functions.

automated function. SA OS/390 automated functions are NetView automation operators, that are assigned to perform specific automation functions. However, SA OS/390 defines its own synonyms, or *automated function names*, for the NetView automation operators, and these function names are referred to in the sample policy databases provided by SA OS/390. For example, the automation operator AUTBASE corresponds to the SA OS/390 automated function BASEOPER.

automated console operations (ACO). The concept (versus a product) of using computers to perform a large subset of tasks ordinarily performed by operators, or assisting operators in performing these tasks.

automatic restart management. An OS/390 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 OS/390.

automatic restart management element name. In MVS 5.2 or later, OS/390 automatic restart management requires the specification of a unique sixteen character name for each address space which registers with it. All automatic restart management policy is defined in terms of the element name, including SA OS/390's interface with it.

automation. The automatic initiation of actions in response to detected conditions or events. SA OS/390 provides automation for OS/390 applications, OS/390 components, and remote systems that run OS/390. SA OS/390 also provides tools that can be used to develop additional automation.

automation agent. In SA OS/390, 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 control file (ACF). In SA OS/390, a file that contains system-level automation policy information. There is one master automation control file for each NetView system on which SA OS/390 is installed. The SA OS/390 customization dialog must be used to build the automation control files. They must not be edited manually.

automation flags. In SA OS/390, 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 OS/390 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 dialog.

automation manager. In SA OS/390, the automation function is split up between the automation manager and teh 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 stand-alone system.

Automation NetView. In SA OS/390 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 OS/390 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, OS/390 subsystems, OS/390 data sets, and OS/390 components.

automation policy settings. The automation policy information contained in the automation control file. This information is entered using the customization dialog. You can display or modify these settings using the customization dialog.

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 the NetView program.

automation status file. In SA OS/390, a file containing status information for each automated subsystem, component or data set. This information is used by SA OS/390 automation when taking action or when determining what action to take.

automation table. Also called NetView message automation table. See *NetView message automation table*

autotask. See *automation operator*.

available. In VTAM programs, pertaining to a logical unit that is active, connected, enabled, and not at its session limit.

B

basic mode. A central processor mode that does not use logical partitioning. Contrast with *logically partitioned (LPAR) mode*.

BCP Internal Interface. Processor function of CMOS-390, zSeries processor families. It allows the communication between basic control programs such as z/OS and the processor support element in order to exchange information or to perform processor control functions. Programs using this function can perform hardware operations such as ACTIVATE or SYSTEM RESET.

beaconing. The repeated transmission of a frame or messages (beacon) by a console or workstation upon detection of a line break or outage.

C

central processor (CP). The part of the computer that contains the sequencing and processing facilities for instruction execution, initial program load (IPL), and other machine operations.

central processor complex (CPC). A physical collection of hardware that consists of central storage, one or more central processors, timers, and channels.

central site. In a distributed data processing network, the central site is usually defined as the focal point for alerts, application design, and remote system management tasks such as problem management.

CFR/CFS and ISC/ISR. I/O operations can display and return data about integrated system channels (ISC) connected to a coupling facility and coupling facility receiver (CFR) channels and coupling facility sender (CFS) channels.

channel. A path along which signals can be sent; for example, data channel, output channel. See also *link*.

channel path identifier. A system-unique value assigned to each channel path

CHPID. In SA OS/390, channel path ID; the address of a channel.

CHPID port. A label that describes the system name, logical partitions, and channel paths.

channel-attached. (1) Attached directly by I/O channels to a host processor (for example, a channel-attached device). (2) Attached to a controlling unit by cables, rather than by telecommunication lines. Contrast with *link-attached*. Synonymous with *local*.

CI. Console integration.

CICS/VS. Customer Information Control System for Virtual Storage.

CLIST. Command list.

clone. A set of definitions for application instances which 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 the NetView program 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 OS/390, REXX command lists that can be used for automation procedures.

command procedure. In the NetView program, either a command list or a command processor.

command processor. A module designed to perform a specific function. Command processors, which can be written in assembler or a high-level language (HLL), are issued as commands.

Command Tree/2. An OS/2-based program that helps you build commands on an OS/2 window, then routes the commands to the destination you specify (such as a 3270 session, a file, a command line, or an application program). It provides the capability for operators to build commands and route them to a specified destination.

common commands. The SA OS/390 subset of the CPC operations management commands.

common routine. One of several SA OS/390 programs that perform frequently used automation functions. Common routines can be used to create new automation procedures.

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*.

communications path. A set of communications facilities that SA OS/390 uses to connect a focal point system to a target system. There are two types: a NetView connection and a workstation.

communications task. An SA OS/390 task responsible for all communications with a number of workstations. Communications tasks receive commands from target control tasks and send inbound messages to the message monitoring task. There can be many communications tasks. Communications tasks are defined using the configuration dialogs.

configuration dialogs. A user interface for entering the information that defines the SA OS/390 configuration. In SA OS/390 the configuration dialogs are an ISPF application.

connectivity view. In SA OS/390, 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 OS/390, 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 which if supported by an operating system, allows operating system messages to be transferred through an internal hardware interface for display on a system console. Conversely, it allows operating system commands entered at a system console to be transferred through an internal hardware interface to the operating system for processing.

consoles. Workstations and 3270-type devices that manage your enterprise.

Control units. Hardware units that control I/O operations for one or more devices. You can view information about control units through I/O operations, and can start or stop data going to them by blocking and unblocking ports.

controller. A unit that controls I/O operations for one or more devices.

couple data set. A data set that is created through the XCF couple data set format utility and, depending on its designated type, is shared by some or all of the OS/390 systems in a sysplex. See also *sysplex couple data set* and *XCF couple data set*

coupling facility. The hardware element that provides high-speed caching, list processing, and locking functions in a sysplex.

CP. Central processor.

CPC. Central processor complex.

CPC operations management commands. A set of commands and responses for controlling the operation of System/390 CPCs.

CPC subset. All or part of a CPC. It contains the minimum *resource* to support a single control program.

CPCB. Command processor control block; an I/O operations internal control block that contains information about the command being processed.

CPU. Central processing unit. Deprecated term for *processor*.

cross-system coupling facility (XCF). XCF is a component of OS/390 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 dialog. The customization dialog is an ISPF application. They are used to customize the enterprise policy, like for example the automated enterprise resources and the relationships between resources, or the automation policy for systems in the enterprise.

CVC. A channel operating in converted (CVC) mode transfers data in blocks and a CBY channel path transfers data in bytes. Converted CVC or CBY channel paths can communicate with a parallel control unit. This resembles a point-to-point parallel path and dedicated connection, regardless whether it passes through a switch.

D

DASD. Direct access storage device.

data services task (DST). The NetView subtask that gathers, records, and manages data in a VSAM file or a network device that contains network management information.

data set. The major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access.

data set members. Members of partitioned data sets that are individually named elements of a larger file that can be retrieved by name.

DBCS. Double-byte character set.

DCCF. Disabled console communication facility.

DCF. Document composition facility.

Devices. You can see information about all devices (such as printers, tape or disk drives, displays, or communications controllers) attached to a particular switch, and control paths and jobs to devices.

direct access storage device (DASD). A device in which the access time is effectively independent of the location of the data; for example, a disk.

disabled console communication facility (DCCF). An OS/390 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).

distribution manager. The component of the NetView program that enables the host system to use, send, and delete files and programs in a network of computers.

domain. (1) An access method and its application programs, communication controllers, connecting lines, modems, and attached workstations. (2) In SNA, a system services control point (SSCP) and the physical units (PUs), logical units (LUs), links, link stations, and associated resources that the SSCP can control by means of activation requests and deactivation requests.

double-byte character set (DBCS). A character set, such as Kanji, in which each character is represented by a 2-byte code.

DSIPARM. This file is a collection of members of NetView's DSIPARM data set. It determines how automation is applied to resources.

DST. Data Services Task.

E

EBCDIC. Extended binary-coded decimal interchange code. A coded character set consisting of 8-bit coded characters.

ECB. Event control block. A control block used to represent the status of an event.

EMCS. Extended multiple console support.

enterprise. An organization, such as a business or a school, that uses data processing.

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 OS/390 should stop trying to restart or recover an application, subsystem or component, or off-load a data set.

ESA. Enterprise Systems Architecture.

event. (1) In the NetView program, a record indicating irregularities of operation in physical elements of a network. (2) An occurrence of significance to a task; for example, the completion of an asynchronous operation, such as an input/output operation. (3) Events are part of a trigger condition, in a way that if all events of a trigger condition have occurred, a *STARTUP* or *SHUTDOWN* of an application is performed.

exception condition. An occurrence on a system that is a deviation from normal operation. SA OS/390 monitoring highlights exception conditions and allows an SA OS/390 enterprise to be managed by exception.

extended recovery facility (XRF). A facility that minimizes the effect of failures in OS/390, 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*

file manager commands. A set of SA OS/390 commands that read data from or write data to the automation control file. These commands are useful in the development of automation that uses SA OS/390 facilities.

focal point. In the NetView program, 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) **SA OS/390 automation focal point system.** The SA OS/390 automation focal point system is an SA OS/390 NetView system that collects status information from other SA OS/390 NetViews within your enterprise. It is supported by SA OS/390, which uses NetView NNT/OST sessions to forward information to it. (3) **SA OS/390 configuration focal point system.** The SA OS/390 configuration focal point is the system on which SA OS/390 is installed. The SA OS/390 policy databases exist on this system and are built into automation control file fragments, and processor control file on this system (if this is being done using SA OS/390). (4) **SA OS/390 workstation focal-system.** The SA OS/390 workstation focal point must be the same as the NMC focal point system. SA OS/390 code is installed on both the workstation and the NMC focal point system. The SA OS/390 workstation focal point system collects status information about systems and applications within your enterprise. It is supported by SA OS/390. (5) **NMC focal point system**. The NMC focal point system is a NetView system with an attached workstation server and LAN which 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. (6) **NPDA focal point system.** This is a NetView system which collects all the NPDA alerts that are generated within your enterprise. It is supported by NetView. If you have SA OS/390 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*. (7) **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. (8) **SA OS/390 focal point system.** This is a NetView system that has SA OS/390 host code installed, and network connections to a number of SA OS/390 workstations. The SA OS/390 focal point system receives messages from the systems and operator consoles of the machines 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. (9) **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 the NetView program, 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 OS/390 outbound gateway operator logs onto the other NetView session without human operator intervention. Each end of a gateway session has both an inbound and outbound gateway operator.

generic alert. Encoded alert information that uses code points (defined by IBM and possibly customized by users or application programs) stored at an alert receiver, such as NetView.

generic routines. In SA OS/390, a set of self-contained automation routines that can be called from the NetView message automation table, or from user-written automation procedures.

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 be of type STANDARD or SYSPLEX.

H

Hardware Management Console (HMC). A console used to monitor and control hardware such as CMOS-390 or zSeries processors.

Hardware Management Console Application (HWMCA). A direct-manipulation object-oriented graphical user interface that provides single point of control and single system image for hardware elements. HWMCA provides customer grouping support, aggregated and real-time system status using colors, consolidated hardware messages support, consolidated operating system messages support, consolidated service support, and hardware commands targeted at a single system, multiple systems, or a customer group of systems.

heartbeat. In SA OS/390, a function which monitors the validity of the status forwarding path between remote systems and the NMC focal point system, and monitors the availability of remote OS/390 systems, to ensure that status information displayed on the SA OS/390 workstation is current.

help panel. An online panel that tells you how to use a command or another aspect of a product.

hierarchy. In the NetView program, the resource types, display types, and data types that make up the organization, or levels, in a network.

high-level language (HLL). A programming language that does not reflect the structure of any particular computer or operating system. For the NetView program, the high-level languages are PL/I and C.

HLL. High-level language.

host system. In a coupled system or distributed system environment, the system on which the facilities for centralized automation run. SA OS/390 publications refer to target systems or focal-point systems instead of hosts.

host (primary processor). The processor at which you enter a command (also known as the *issuing processor*)

HWMCA. Hardware Management Console Application. Application for the graphic hardware management console that monitors and controls a central processor complex. It is attached to a target processor (a system 390 microprocessor cluster) as a dedicated system console. This microprocessor uses OCF to process commands.

I

images. A grouping of processors and I/O devices that you define. You can define a single-image mode which allows a multiprocessor system to function as one central processor image.

IMS/VS. Information Management System/Virtual Storage.

inbound. In SA OS/390, messages sent to the focal-point system from the PS/2 computer or target system.

inbound gateway operator. The automation operator that receives incoming messages, commands, and responses from the outbound gateway operator at the sending system. The inbound gateway operator handles communications with other systems using a gateway session.

Information Management System/Virtual Storage (IMS/VS). A database/data communication (DB/DC) system that can manage complex databases and networks. Synonymous with IMS.

INGEIO PROC. The I/O operations default procedure name; part of the SYS1.PROCLIB

initial program load (IPL). (1) The initialization procedure that causes an operating system to commence operation. (2) The process by which a configuration image is loaded into storage at the beginning of a workday or after a system malfunction. (3) The process of loading system programs and preparing a system to run jobs.

initialize automation. SA OS/390-provided automation that issues the correct OS/390 start command for each subsystem when SA OS/390 is initialized. The automation ensures that subsystems are started in the order specified in the automation control file and that prerequisite applications are functional.

input/output support processor (IOSP). The hardware unit that provides I/O support functions for the primary support processor and maintenance support functions for the processor controller.

Interactive System Productivity Facility (ISPF). An IBM licensed program that serves as a full-screen editor and dialog manager. Used for writing application programs, it provides a means of generating standard screen panels and interactive dialogs between the application programmer and the terminal user.

interested operator list. The list of operators who are to receive messages from a specific target system.

internal token. A *logical token* (LTOK); name by which the I/O resource or object is known; stored in IODF.

IOCDS. I/O configuration data set. The data set that describes the I/O configuration.

I/O Ops. See *I/O operations*

IOSP. Input/Output Support Processor.

I/O operations. The part of SA OS/390 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).

I/O resource number. Combination of channel path identifier (CHPID), device number, etc. See internal token.

IPL. Initial program load.

ISA. Industry Standard Architecture.

ISPF. Interactive System Productivity Facility.

ISPF console. From this 3270-type console you are logged onto ISPF to use the runtime panels for I/O operations and SA OS/390 customization panels.

issuing host. See *primary host*; the base program at which you enter a command for processing.

J

JCL. Job control language.

JES. Job entry subsystem.

job. (1) A set of data that completely defines a unit of work for a computer. A job usually includes all necessary computer programs, linkages, files, and instructions to the operating system. (2) An address space.

job control language (JCL). A problem-oriented language designed to express statements in a job that are used to identify the job or describe its requirements to an operating system.

job entry subsystem (JES). A facility for spooling, job queuing, and managing I/O. In SA OS/390 publications, JES refers to JES2 or JES3, unless distinguished as being either one or the other.

K

Kanji. An ideographic character set used in Japanese. See also *double-byte character set*

L

LAN. Local area network.

line mode. A form of screen presentation in which the information is presented a line at a time in the message area of the terminal screen. Contrast with *full-screen mode*.

link. (1) In SNA, the combination of the link connection and the link stations joining network nodes; for example, a System/370 channel and its associated protocols, a serial-by-bit connection under the control of synchronous data link control (SDLC). (2) In SA OS/390, link connection is the physical medium of transmission.

link-attached. Describes devices that are physically connected by a telecommunication line. Contrast with *channel-attached*.

local. Pertaining to a device accessed directly without use of a telecommunication line. Synonymous with *channel-attached*.

local area network (LAN). (1) A network in which a set of devices is connected for communication. They can be connected to a larger network. See also *token ring*. (2) A network in which communications are limited to a moderately sized geographic area such as a single office building, warehouse, or campus, and which do not generally extend across public rights-of-way.

logical partition (LP). A subset of the processor hardware that is defined to support an operating system. See also *logically partitioned (() mode*.

logical switch number (LSN). Assigned with the switch parameter of the CHPID macro of the IOCP

logical token (LTOK). Resource number of an object in the IODF

logical unit (LU). In SNA, a port through which an end user accesses the SNA network and the functions provided by system services control points (SSCPs). An LU can support at least two sessions — one with an SSCP and one with another LU — and may be capable of supporting many sessions with other LUs. See also *physical unit (PU)* and *system services control point (SSCP)*.

logical unit (LU) 6.2. A type of logical unit that supports general communications between programs in a distributed processing environment. LU 6.2 is characterized by (a) a peer relationship between session partners, (b) efficient use of a session for multiple transactions, (c) comprehensive end-to-end error processing, and (d) a generic application program interface (API) consisting of structured verbs that are mapped into a product implementation. Synonym for advanced program-to-program communications (APPC).

logically partitioned (LPAR) mode. A central processor mode that enables an operator to allocate system processor hardware resources among several logical partitions. Contrast with *basic mode*.

LOGR. The sysplex logger.

LP. Logical partition.

LPAR. Logically partitioned (mode).

LU. Logical unit.

LU-LU session. In SNA, a session between two logical units (LUs) in an SNA network. It provides communication between two end users, or between an end user and an LU services component.

LU 6.2. Logical unit 6.2.

LU 6.2 session. A session initiated by VTAM on behalf of an LU 6.2 application program, or a session initiated by a remote LU in which the application program specifies that VTAM is to control the session by using the APPCCMD macro.

M

MCA. Micro Channel* architecture.

MCS. Multiple console support.

member. A specific function (one or more modules/routines) of a multisystem application that is defined to XCF and assigned to a group by the multisystem application. A member resides on one system in the sysplex and can use XCF services to communicate (send and receive data) with other members of the same group.

NetView message automation table. A table against which the NetView program compares incoming messages. A match with an entry triggers the specified response. SA OS/390 entries in the NetView automation table trigger an SA OS/390 response to target system conditions.

message class. A number that SA OS/390 associates with a message to control routing of the message. During automated operations, the classes associated with each message issued by SA OS/390 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 OS/390 process of sending messages generated at an SA OS/390 target system to the SA OS/390 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). An OS/390 table that screens all messages sent to the OS/390 console. The MPF compares these messages with a customer-defined list of messages on which to automate, suppress from the OS/390 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 OS/390 console.

Micro Channel architecture. The rules that define how subsystems and adapters use the Micro Channel bus in a computer. The architecture defines the services that each subsystem can or must provide.

microprocessor. A processor implemented on one or a small number of chips.

migration. Installation of a new version or release of a program to replace an earlier version or release.

MP. Multiprocessor.

MPF. Message processing facility.

Multiple Virtual Storage (MVS). An IBM licensed program. MVS, which is the predecessor of OS/390, is an operating system that controls the running of programs on a System/390 or System/370 processor. MVS includes an appropriate level of the Data Facility Product (DFP) and Multiple Virtual Storage/Enterprise Systems Architecture System Product Version 5 (MVS/ESA SP5)

multiprocessor (MP). A CPC that can be physically partitioned to form two operating processor complexes.

multisystem application. An application program that has various functions distributed across OS/390 images in a multisystem environment.

multisystem environment. An environment in which two or more OS/390 images reside in one or more processors, and programs on one image can communication with programs on the other images.

MVS. Multiple Virtual Storage, predecessor of OS/390.

MVS image. A single occurrence of the MVS/ESA operating system that has the ability to process work.

MVS/JES2. Multiple Virtual Storage/Job Entry System 2. An OS/390 subsystem that receives jobs into the system, converts them to internal format, selects them for execution, processes their output, and purges them from the system. In an installation with more than one processor, each JES2 processor independently controls its job input, scheduling, and output processing.

MVS/ESA. Multiple Virtual Storage/Enterprise Systems Architecture.

MVS/ESA SP. Multiple Virtual Storage/Enterprise Systems Architecture System Product.

MVS/XA. Multiple Virtual Storage for Extended Architecture.

N

NAU. (1) Network accessible unit. (2) Network addressable unit.

NCCF. Network Communications Control Facility.

NCP. (1) Network Control Program (IBM licensed program). Its full name is Advanced Communications Function for the Network Control Program. Synonymous with *ACF/NCP*. (2) Network control program (general term).

NetView. An IBM licensed program used to monitor a network, manage it, and diagnose network problems. NetView consists of a command facility that includes a presentation service, command processors, automation based on command lists, and a transaction processing structure on which the session monitor, hardware monitor, and terminal access facility (TAF) network management applications are built.

network accessible unit (NAU). A logical unit (LU), physical unit (PU), control point (CP), or system services control point (SSCP). It is the origin or the destination of information transmitted by the path control network. Synonymous with *network addressable unit*.

network addressable unit (NAU). Synonym for *network accessible unit*.

NetView automation procedures. A sequence of commands, packaged as a NetView command list or a command processor written in a high-level language. An automation procedure performs automation functions and runs under the NetView program.

NetView Command list language. An interpretive language unique to NetView that is used to write command lists.

NetView (NCCF) console. A 3270-type console for NetView commands and runtime panels for system operations and processor operations

NetView hardware monitor. The component of NetView that helps identify network problems, such as hardware, software, and microcode, from a central control point using interactive display techniques. Formerly called *network problem determination application*

NetView log. The log in which NetView records events pertaining to NetView and SA OS/390 activities.

NetView-NetView task (NNT). The task under which a cross-domain NetView operator session runs. Each NetView program must have a NetView-NetView task to establish one NNT session. See *operator station task*

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.

NetView-NetView task session. A session between two NetView programs that runs under an NetView-NetView task. In SA OS/390, 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 Communications Control Facility (NCCF). The operations control facility for the network. NCCF consists of a presentation service, command processors, automation based on command lists, and a transaction processing structure on which the network management applications NLDM and NPDA are built. NCCF is a precursor to the NetView command facility.

Network Control Program (NCP). An IBM licensed program that provides communication controller support for single-domain, multiple-domain, and interconnected network capability. Its full name is Advanced Communications Function for the Network Control Program.

Networking NetView. In SA OS/390 the NetView that performs network management functions, such as managing the configuration of a network. In SA OS/390 it is common to also route alerts to the Networking NetView.

Network Problem Determination Application (NPDA). An NCCF application that helps you identify network problems, such as hardware, software, and microcode, from a central control point using interactive display methods. The alert manager for the network. The precursor of the NetView hardware monitor.

NMC focal point system. See *focal point system*

NMC workstation. The NMC workstation is the primary way to dynamically monitor SA OS/390 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.

NIP. Nucleus initialization program.

NNT. NetView-NetView task.

notification message. An SA OS/390 message sent to a human notification operator to provide information about significant automation actions. Notification messages are defined using the customization dialog.

notification operator. A NetView console operator who is authorized to receive SA OS/390 notification messages. Authorization is made through the customization dialog.

NPDA. Network Problem Determination Application.

NPDA focal point system. See *focal point system*

NTRI. NCP/token-ring interconnection.

nucleus initialization program (NIP). The program that initializes the resident control program; it allows the operator to request last-minute changes to certain options specified during system generation.

O

OCA. In SA OS/390, operator console A, the active operator console for a target system. Contrast with *OCB*.

OCB. In SA OS/390, operator console B, the backup operator console for a target system. Contrast with *OCA*.

OCF. Operations command facility.

OCF-based processor. A central processor complex that uses an operations command facility for interacting with human operators or external programs to perform operations management functions on the CPC. Contrast with *screen-oriented processor*.

OPC/A. Operations Planning and Control/Advanced.

OPC/ESA. Operations Planning and Control/Enterprise Systems Architecture.

operating system (OS). Software that controls the execution of programs and that may provide services such as resource allocation, scheduling, input/output control, and data management. Although operating systems are predominantly software, partial hardware implementations are possible. (T)

operations. The real-time control of a hardware device or software function.

operations command facility (OCF). A facility of the central processor complex that accepts and processes operations management commands.

Operations Planning and Control/Advanced (OPC/A). A set of IBM licensed programs that automate, plan, and control batch workload. OPC/A analyzes system and workload status and submits jobs accordingly.

Operations Planning and Control/ESA (OPC/ESA). A set of IBM licensed programs that automate, plan, and control batch workload. OPC/ESA analyzes system and workload status and submits jobs accordingly. The successor to OPC/A.

operator. (1) A person who keeps a system running. (2) A person or program responsible for managing activities controlled by a given piece of software such as OS/390, 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 OS/390, a console that displays output from and sends input to the operating system (OS/390, VM, VSE, or LINUX). Also called *operating system console*. In the SA OS/390 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.

OS/390 component. A part of OS/390 that performs a specific OS/390 function. In SA OS/390, component refers to entities that are managed by SA OS/390 automation.

OS/390 subsystem. Software products that augment the OS/390 operating system. JES and TSO/E are examples of OS/390 subsystems. SA OS/390 includes automation for some OS/390 subsystems.

OS/390 system. An OS/390 image together with its associated hardware, which collectively are often referred to simply as a system, or OS/390 system.

OSA. I/O operations can display the open system adapter (OSA) channel logical definition, physical attachment, and status. You can configure an OSA channel on or off.

OST. Operator station task.

outbound. In SA OS/390, messages or commands from the focal-point system to the PS/2 computer or 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 OS/390 panels you can see status, type commands on a command line using a keyboard, configure your system, and passthru to other consoles. See also *help panel*. (3) In computer graphics, a display image that defines the locations and characteristics of display fields on a display surface. Contrast with *screen*.

parallel channels. Parallel channels operate in either byte (BY) or block (BL) mode. You can change connectivity to a parallel channel operating in block mode.

parameter. (1) A variable that is given a constant value for a specified application and that may denote the application. (2) An item in a menu for which the user specifies a value or for which the system provides a value when the menu is interpreted. (3) Data passed to a program or procedure by a user or another program, namely as an operand in a language statement, as an item in a menu, or as a shared data structure.

partition. (1) A fixed-size division of storage. (2) In VSE, a division of the virtual address area that is available for program processing. (3) On an IBM Personal Computer fixed disk, one of four possible storage areas of variable size; one can be accessed by DOS, and each of the others may be assigned to another operating system.

partitionable CPC. A CPC that can be divided into 2 independent CPCs. See also *physical partition, single-image mode, MP, side*.

partitioned data set (PDS). A data set in direct access storage that is divided into partitions, called *members*, each of which can contain a program, part of a program, or data.

passive monitoring. In SA OS/390, the receiving of unsolicited messages from OS/390 systems and their resources. These messages can prompt updates to resource status displays. See also *active monitoring*

path. Communication link (either NetView or PS/2) between a processor and console

PCE. Processor controller. Also known as the "support processor" or "service processor" in some processor families.

PDS. Partitioned data set.

physical partition. Part of a CPC that operates as a CPC in its own right, with its own copy of the operating system.

physical unit (PU). In SNA, the component that manages and monitors the resources (such as attached links and adjacent link stations) of a node, as requested by a system services control point (SSCP) through an SSCP-PU session. An SSCP activates a session with the physical unit to indirectly manage, through the PU, resources of the node such as attached links.

physically partitioned (PP) configuration. A mode of operation that allows a multiprocessor (MP) system to function as two or more independent CPCs having separate power, water, and maintenance boundaries. Contrast with *single-image (SI) configuration*.

POI. Program operator interface.

policy. The automation and monitoring specifications for an SA OS/390 enterprise. See *SA OS/390 policy*

policy database. The database where the automation policy is recorded.

POR. Power-on reset.

port. (1) System hardware to which the I/O devices are attached. (2) On an ESCON switch, a port is an addressable connection. The switch routes data through the ports to the channel or control unit. Each port has a name that can be entered into a switch matrix, and you can use commands to change the switch configuration. (3) An access point (for example, a logical unit) for data entry or exit. (4) A functional unit of a node through which data can enter or leave a data network. (5) In data communication, that part of a data processor that is dedicated to a single data channel for the purpose of receiving data from or transmitting data to one or more external, remote devices. (6) power-on reset (POR) (7) A function that re-initializes all the hardware in a CPC and loads the internal code that enables the CPC to load and run an operating system.

PP. Physically partitioned (configuration).

PPT. Primary POI task.

primary host. The base program at which you enter a command for processing.

primary POI task (PPT). The NetView subtask that processes all unsolicited messages received from the VTAM program operator interface (POI) and delivers them to the controlling operator or to the command processor. The PPT also processes the initial command specified to execute when NetView is initialized and timer request commands scheduled to execute under the PPT.

primary system. A system is a primary system for an application if the application is normally meant to be running there. SA OS/390 starts the application on all the primary systems defined for it.

problem determination. The process of determining the source of a problem; for example, a program component, machine failure, telecommunication facilities, user or contractor-installed programs or equipment, environment failure such as a power loss, or user error.

processor controller. Hardware that provides support and diagnostic functions for the central processors.

processor operations. The part of SA OS/390 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.

processor operations control file. Named by your system programmer, this file contains configuration and customization information. The programmer records the name of this control file in the processor operations file generation panel ISQDPG01.

processor operations workstation. This 3270-type display echoes commands run by automation and relays commands from the focal point system to the target processor, and messages from the target processor to the focal point system. The display can also be used as operator consoles. The workstation attaches to a processor through the processor controller.

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 *(*.

ProcOps. See *Processor Operations*

product automation. Automation integrated into the base of SA OS/390 for the products DB2, CICS, IMS, OPC (formerly called *features*.

program to program interface (PPI). A NetView function that allows user programs to send or receive data buffers from other user programs and to send alerts to the NetView hardware monitor from system and application programs.

protocol. In SNA, the meanings of, and the sequencing rules for, requests and responses used for

managing the network, transferring data, and synchronizing the states of network components.

proxy resource. A resource defined like an entry type *Application* representing a processor operations target system.

PR/SM. Processor Resource/Systems Manager.

PU. Physical unit.

R

remote system. A system that receives resource status information from an SA OS/390 focal point system. An SA OS/390 remote system is defined as part of the same SA OS/390 enterprise as the SA OS/390 focal point system to which it is related.

requester. A requester is a workstation software, which enables users to log on to a domain, that is, to the server(s) belonging to this domain, and use the resources in this domain. After the log on to a domain, users can access the shared resources and use the processing capability of the server(s). Because the bigger part of shared resources is on the server(s), users can reduce hardware investment.

resource. (1) Any facility of the computing system or operating system required by a job or task, and including main storage, input/output devices, the processing unit, data sets, and control or processing programs. (2) In NetView, any hardware or software that provides function to the network. (3) In SA OS/390, any OS/390 application, OS/390 component, job, device, or target system capable of being monitored or automated through SA OS/390.

Resource Access Control Facility (RACF). A program that can provide data security for all your resources. RACF protects data from accidental or deliberate unauthorized disclosure, modification, or destruction.

resource group. A physically partitionable portion of a processor. Also known as a *side*.

Resource Monitoring Facility (RMF). A program that measures and reports on the availability and activity of system hardware and software resources, such as processors, devices, storage, and address spaces. RMF can issue reports about system performance problems as they occur.

Resource Object Data Manager (RODM). A data cache manager designed to support process control and automation applications. RODM provides an in-memory data cache for maintaining real-time data in an address space that is accessible by multiple applications. RODM also allows an application to query an object and receive a rapid response and act on it.

resource token. A unique internal identifier of an ESCON resource or resource number of the object in the IODF

restart automation. SA OS/390-provided automation that monitors subsystems to ensure that they are running. If a subsystem fails, SA OS/390 attempts to restart it according to the policy in the automation control file.

Restructured Extended Executor (REXX). An interpretive language used to write command lists.

return code. A code returned from a program used to influence the issuing of subsequent instructions.

REXX. Restructured Extended Executor.

REXX procedure. A command list written with the Restructured Extended Executor (REXX) which is an interpretive language.

RMF. Resource Measurement Facility.

RODM. Resource Object Data Manager.

S

SAF. Security Authorization Facility.

SA OS/390 automation focal point system. See *focal point system*

SA OS/390 customization dialog. An ISPF application through which the SA OS/390 policy administrator defines policy for individual OS/390 systems and builds automation control data.

SA OS/390 customization focal point system. See *focal point system*

SA OS/390 enterprise. The group of systems and resources defined in the customization dialog under one enterprise name. An SA OS/390 enterprise consists of connected OS/390 systems running SA OS/390.

SA OS/390 focal point system. See *focal point system*.

SA OS/390 policy. The description of the systems and resources that make up an SA OS/390 enterprise, together with their monitoring and automation definitions.

SA OS/390 policy administrator. The member of the operations staff who is responsible for defining SA OS/390 policy.

SA OS/390 satellite. If you are running two NetViews on an OS/390 system to split the automation and networking functions of NetView, it is common to route alerts to the Networking NetView. For SA OS/390 to process alerts properly on the Networking NetView,

you must install a subset of SA OS/390 code, called an *SA OS/390 satellite* on the Networking NetView.

SCA. In SA OS/390, system console A, the active system console for a target hardware. Contrast with *SCB*.

SCB. In SA OS/390, system console B, the backup system console for a target hardware. Contrast with *SCA*.

screen. Deprecated term for display panel.

screen handler. In SA OS/390, software that interprets all data to and from a full-screen image of a target system. The interpretation depends on the format of the data on the full-screen image. Every processor and operating system has its own format for the full-screen image. A screen handler controls one PS/2 connection to a target system.

SDF. Status display facility.

SDLC. Synchronous data link control.

SDSF. System Display and Search Facility.

secondary system. A system is a secondary system for an application if it is defined to automation on that system, but the application is not normally meant to be running there. Secondary systems are systems to which an application can be moved in the event that one or more of its primary systems are unavailable. SA OS/390 does not start the application on its secondary systems.

server. A server is a workstation, that shares resources, which include directories, printers, serial devices, and computing powers.

service period. Service periods allow the users to schedule the availability of applications. A service period is a set of time intervals (service windows), during which an application should be active.

service threshold. An SA OS/390 policy setting that determines when to notify the operator of deteriorating service for a resource. See *alert threshold* and *warning threshold*

service language command (SLC). The line-oriented command language of processor controllers or service processors.

service processor (SVP). The name given to a processor controller on smaller System/370 processors.

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 OS/390-provided automation that manages the shutdown process for subsystems by issuing shutdown commands and responding to prompts for additional information.

side. A part of a partitionable CPC that can run as a physical partition and is typically referred to as the A-side or the B-side.

Simple Network Management Protocol (SNMP) . An IP based industry standard protocol to monitor and control resources in an IP network.

single image. A processor system capable of being physically partitioned that has not been physically partitioned. Single-image systems can be target hardware processors.

single-image (SI) mode. A mode of operation for a multiprocessor (MP) system that allows it to function as one CPC. By definition, a uniprocessor (UP) operates in single-image mode. Contrast with *physically partitioned (PP) configuration*.

SLC. Service language command.

SMP/E. System Modification Program Extended.

SNA. Systems Network Architecture.

SNA network. In SNA, the part of a user-application network that conforms to the formats and protocols of systems network architecture. It enables reliable transfer of data among end users and provides protocols for controlling the resources of various network configurations. The SNA network consists of network addressable units (NAUs), boundary function components, and the path control network.

SNMP. Simple Network Management Protocol (a TCP/IP protocol). A protocol that allows network management by elements, such as gateways, routers, and hosts. This protocol provides a means of communication between network elements regarding network resources.

solicited message. An SA OS/390 message that directly responds to a command. Contrast with *unsolicited message*.

SRPI. Server-Requester Programming Interface.

SSCP. System services control point.

SSI. Subsystem interface.

start automation. SA OS/390-provided automation that manages and completes the startup process for subsystems. During this process, SA OS/390 replies to prompts for additional information, ensures that the startup process completes within specified time limits, notifies the operator of problems, if necessary, and brings subsystems to an UP (or ready) state.

startup. The point in time at which a subsystem or application is started.

status. The measure of the condition or availability of the resource.

status focal point system. See *focal point system*

status display facility (SDF). The system operations part of SA OS/390 that displays status of resources such as applications, gateways, and write-to-operator messages (WTORs) on dynamic color-coded panels. SDF shows spool usage problems and resource data from multiple systems.

steady state automation. The routine monitoring, both for presence and performance, of subsystems, applications 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 OS/390 to map and manage storage on a coupling facility. See cache structure, list structure, and lock structure.

subgroup. A named set of systems. A subgroup is part of an SA OS/390 enterprise definition and is used for monitoring purposes.

SubGroup entry. A construct, created with the customization dialog, used to represent and contain policy for a subgroup.

subsystem. (1) A secondary or subordinate system, usually capable of operating independent of, or asynchronously with, a controlling system. (2) In SA OS/390, an OS/390 application or subsystem defined to SA OS/390.

subsystem interface. The OS/390 interface over which all messages sent to the OS/390 console are broadcast.

support element. A hardware unit that provides communications, monitoring, and diagnostic functions to a central processor complex (CPC).

support processor. Another name given to a processor controller on smaller System/370 processors; see *service processor*.

SVP. Service processor.

switches. ESCON directors are electronic units with ports that dynamically switch to route data to I/O devices. The switches are controlled by I/O operations commands that you enter on a workstation.

switch identifier. The switch device number (swchdevn), the logical switch number (LSN) and the switch name

symbolic destination name (SDN). Used locally at the workstation to relate 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 non-switched links. The configuration of the link connection may be point-to-point, multi-point, 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.

SysOps. See *System Operations*

sysplex. A set of OS/390 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, OS/390 provides the coupling services that handle the messages, data, and status for the parts of a multisystem application that has its workload spread across two or more of the connected processors, sysplex timers, coupling facilities, and couple data sets (which contains policy and states for automation).

A parallel sysplex is a sysplex which 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 OS/390 systems in a sysplex must have connectivity to the sysplex couple data set. See also *couple data set*

Sysplex Timer. An IBM unit that synchronizes the time-of-day (TOD) clocks in multiple processors or processor sides. External Time Reference (ETR) is the OS/390 generic name for the IBM Sysplex Timer (9037).

system. In SA OS/390, system means a focal point system (OS/390) or a target system (MVS, VM, VSE, TPF or CF).

System Automation for OS/390. The full name for SA OS/390.

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 OS/390, the hardware system console for processor controllers or service processors of processors connected using SA OS/390. In the SA OS/390 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 OS/390 you can select SDSF from a pull-down menu to see the resources' status, view the OS/390 system log, see WTOR messages, and see active jobs on the system.

System entry type. A construct, created with the customization dialog, 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 OS/390 system.

system operations. The part of SA OS/390 that monitors and controls system operations applications and subsystems such as NetView, SDSF, JES, RMF, TSO, RODM, ACF/VTAM, CICS, IMS, and OPC.

system services control point (SSCP). In SNA, the focal point within an SNA network for managing the configuration, coordinating network operator and problem determination requests, and providing directory support and other session services for end users of the network. Multiple SSCPs, cooperating as peers, can divide the network into domains of control, with each SSCP having a hierarchical control relationship to the physical units and logical units within its domain.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of, networks.

System/390 microprocessor cluster. A configuration that consists of central processor complexes (CPCs) and may have one or more integrated coupling facilities.

T

TAF. Terminal access facility.

target. A processor or system monitored and controlled by a focal point system.

target control task. In SA OS/390, 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 OS/390, 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 OS/390, 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 OS/390 tasks are NetView tasks. See also *communications task*, *message monitor task*, and *target control task*.

telecommunication line. Any physical medium, such as a wire or microwave beam, that is used to transmit data.

terminal access facility (TAF). (1) A NetView function that allows you to log onto multiple applications either on your system or other systems. You can define TAF sessions in the SA OS/390 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 OS/390 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 OS/390, MVS/ESA, and MVS/XA.

Time-Sharing Option/Extended (TSO/E). An option of OS/390 that provides conversational time-sharing from remote terminals. TSO/E allows a wide variety of users to perform many different kinds of tasks. It can handle short-running applications that use fewer sources as well as long-running applications that require large amounts of resources.

timers. A NetView command that issues a command or command processor (list of commands) at a specified time or time interval.

TOD. Time of day.

token ring. A network with a ring topology that passes tokens from one attaching device to another; for example, the IBM Token-Ring Network product.

TP. Transaction program.

transaction program. In the VTAM program, a program that performs services related to the processing of a transaction. One or more transaction programs may operate within a VTAM application program that is using the VTAM application program interface (API). In that situation, the transaction program would request services from the applications program using protocols defined by that application program. The application program, in turn, could request services from the VTAM program by issuing the APPCCMD macro instruction.

transitional automation. the actions involved in starting and stopping subsystems and applications which have been defined to SA OS/390. 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. On top of requests, triggers, in combination with events, are used to control the starting and stopping of applications in a single system or a Parallel Sysplex. Triggers act as inhibitors for the requested action.

TSO. Time Sharing Option.

TSO console. From this 3270-type console you are logged onto TSO or ISPF to use the runtime panels for I/O operations and SA OS/390 customization panels.

TSO/E. TSO Extensions.

U

UCB. The unit control block; an MVS/ESA data area that represents a device and which is used for allocating devices and controlling I/O operations.

unsolicited message. An SA OS/390 message that is not a direct response to a command. Contrast with *solicited message*.

user task. An application of the NetView program defined in a NetView TASK definition statement.

V

Virtual Machine/System Product (VM/SP). An IBM licensed program. It is an operating system that manages the resources of a real processor to provide virtual machines to end users. As a time-sharing system control program, it consists of the virtual machine control program (CP), the conversational monitor system (CMS), the group control system (GCS), and the interactive problem control system (IPCS).

Virtual Storage Extended (VSE). An IBM licensed program whose full name is Virtual Storage Extended/Advanced Function. It is an operating system that controls the execution of programs.

Virtual Telecommunications Access Method (VTAM). An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability. Its full name is Advanced Communications Function for the Virtual Telecommunications Access Method. Synonymous with *ACF/VTAM*.

VM/ESA*. Virtual Machine/Enterprise Systems Architecture*

VM/SP. Virtual Machine/System Product.

VM/SP HPO. Virtual Machine/System Product High Performance Option.

VSE. Virtual Storage Extended.

VTAM. Virtual Telecommunications Access Method.

W

warning threshold. An application service value that determines the level at which SA OS/390 changes the associated icon on the NMC workstation to the warning color. See *alert threshold*

workstation. In SA OS/390 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 OS/390 operator console. This request is made by an application and is handled by the WTO processor, which is part of the OS/390 supervisor program.

write-to-operator-with-reply (WTOR). A request to send a message to an operator at the OS/390 operator console which 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 OS/390 supervisor program.

WTO. Write-to-Operator.

WTOR. Write-to-Operator-with-Reply.

WWV. The US National Institute of Standards and Technology (NIST) radio station that provides standard time information. A second station, known as WWVB, provides standard time information at a different frequency.

X

XCF. Cross-system coupling facility.

XCF couple data set. The name for the sysplex couple data set prior to MVS/ESA System Product Version 5 Release 1. See *sysplex couple data set*

XCF group. A set of related members that a multisystem application defines to XCF. A member is a specific function, or instance, of the application. A member resides on one system and can communicate with other members of the same group across the sysplex.

XRF. Extended recovery facility.

Numerics

3270 connection card. A 3270 emulator card installed in a workstation used to connect the workstation to the target system. Four 3270 connection cards can be installed in a workstation.

37x5. A 3705, 3725, or 3745 communication controller.

3x74. A 3174 or 3274 subsystem control unit. A control unit to which 3270-family display stations are attached.

Index

Special characters

&SUBSASID
  usage of 41
&SUBSUSSJOB
  usage of 41

A

abnormal end messages 18
active connector 52
ACTIVE messages 17
ACTIVMSG 17, 23
adding a message to automation 83
adding an application to automation 21
adding an entry to the NetView message automation table 85
address spaces
  obtaining unique jobnames 42
advanced automation options
  exits 127
  external global variables 141, 149
alternate CDS 49
  turn into primary CDS 49
alternate CDS recovery
  customization of 50
alternate couple data set
  specifying 66
AOCMSG call 104
AOCMSG generic routine 98
AOCQRY common routine
  automation availability 98
  message automation 112
AOCTRACE
  use in testing 107
  use in traces 110
AOCUPDT common routine
  and the AOFEXSTA exit 128
  to update status information 98
AOF_ASSIGN_JOBNAME 141, 149
AOF_EMCS_AUTOTASK_... 141, 149
AOF_EMCS_CN_ASSIGNMENT 141, 149
AOF_GATOPER_PW_CHANGE 142, 150
AOF_INIT_MCSFLAG 142, 150
AOF_INIT_ROUTCDE 142, 151
AOF_INIT_SYSCONID 142, 151
AOF_NETWORK_DOMAIN_ID 142
AOF_PRODLVL 143
AOF_SET_AVM_RESTART_EXIT 144, 152
AOF.0DEBUG 141
AOF.0TRACE 141
AOF3WTIME 144, 153
AOFACFINIT 141, 149
AOFADMON 93
AOFAJMON 93
AOFAOCCLONE 141
AOFAPMON 93
AOFARMPPI 11
AOFATMON 93
AOFCNMASK 141, 150
AOFCONFIRM global variable 136
AOFCPSM 93
AOFCTLOPT 141, 149
AOFDEBUG 141
AOFDEBUG global variable 110
AOFDEFAULT_TARGET 141, 150
AOFDOM 3
AOFEXC00 exit 129
AOFEXC01 exit 129
AOFEXC03 exit 131
AOFEXC04 exit 132
AOFEXC2 exit 130
AOFEXDEF 33
AOFEXDEF exit 126, 149
AOFEXI01 exit 127
AOFEXI02 exit 127
AOFEXI03 exit 127
AOFEXI04 exit 127
AOFEXINT exit 127, 137, 144, 152
AOFEXPLAIN_USER 142, 150
AOFEXSTA exit 128
AOFGMFHSWAIT 11
AOFHOLDA 4
AOFHOLDD 4
AOFHOLDE 5
AOFHOLDI 4
AOFHOLDW 5
AOFINITIALSTARTTYP 142
AOFINITOPOCMD 13
AOFINITREPLY 143, 150
AOFJES3JOB 5
AOFJESPREFX 143
AOFLOCALHOLD 143, 151
AOFLOPAUTOx 5
AOFMINORCHK 143, 151
AOFMOVOPT 143, 151
AOFMSGST 3
AOFMSGSY 3
AOFNORMA 6
AOFNORMD 6
AOFNORME 7
AOFNORMI 6
AOFNORMW 7
AOFOPAUTO1 8
AOFOPAUTO2 8
AOFOPBASEOPER 8
AOFOPGATOPER 11
AOFOPGSSOPER 10
AOFOPJESOPER 9
AOFOPMONOPER 9
AOFOPMSGOPER 9
AOFOPNETOPER 9
AOFOPRECOPER 10
AOFOPSHUTOPER 10
AOFOPSYSOPER 8
AOFOPTOPOMGR 13
AOFOPWTORS 11
AOFPAUSE 143, 151
AOFQUICKWTOR 143, 152
AOFRELOADOPT 143, 152
AOFRESTARTALWAYS 143, 152
AOFRPCWAIT 143, 152
AOFRSUSS 33
AOFSENDGMFHSREQUEST 144
AOFSERXINT 144, 152
AOFSETSTATEOVERRIDE 144, 152
AOFSETSTATESCOPE 144, 153
AOFSETSTATESTART 144, 153
AOFSHUTCHK 144
AOFSHUTDELAY 144, 153
AOFSHUTOVERRIDE 144, 153
AOFSHUTSCOPE 144
AOFSIRTASK 11
AOFSPOOLFULLCMD 144, 153
AOFSPOOLSHORTCMD 144, 153
AOFSUBSYS 144
AOFSYS 3
AOFSYSNAME 144
AOFSYSTEM 145
AOFTDDF task 108
AOFTECMODE 12
AOFTECPPI 12
AOFTECTASK 12
AOFTECTASKQ 12
AOFUSSWAIT 153
  setting 33
AOFUXMON 33, 93
  defining with customization dialogs 38
AOFVTAMJOB 7
application
  adding to automation 21
application messages, entries in MPF list 23
application monitor status 93
application monitoring 93
application type IMAGE
  defining 71
ASCB chaining
  and global variables 149
ASFUSER command 112
assist mode
  for testing automation procedures 108
  monitoring automation with interactive 109
  overview 107
automating
  auxiliary storage shortage recovery 76
  enqueues, long running 74
  long running enqueues 74
  message IXC102A 72
  message IXC402D 72
  USS resources 29
automating processor operations controlled resources 115
automation
  adding an application 21
  advanced functions 149

automation *(continued)*
assigning a message 90
enhanced Parallel Sysplex, enabling 49
extending 95
of IXC102A message 57
of IXC402D message 57
response to messages 83
SYSLOG message 55
automation configuration 22
automation control file 22
defining SDF 172
reload action exit 136
reload permission exit 136
automation flag
global 16
automation flags 15
checking 16
example for using 15
for minor resources 15
with individual messages 15
with status changes 15
automation manager configuration file 22
automation manager global automation flag 16
automation procedure
example 104
automation procedures
calling 95
creating 95
description 95
developing messages 103
external code 99
initializing 97
installing 107
making generic 102
structure 96
testing and debugging 107
using AOCTRACE 110
writing your own 95
automation programmer xiii
automation status file 175
coding your own information 112
using commands 99
automation table
generic entries 17
structure 1
automation table statements
user-written 13
automation tables
abend messages 18
AOFMSGSY 3
fragments 3
integrating 13
master automation tables 2
message addition 83
samples 1
status transition messages 17
auxiliary storage shortage
customization of recovery 58
recovery 58
auxiliary storage shortage recovery
automating 76
defining the handling of jobs 77
defining the local page data set 77

B

BASEOPER 152
BLDVIEWS 27
BPX.JOBNAME 33

C

cascades 8
CDEMATCH common routine 112
CDS
INGPLEX command 50
CDS (couple data set) 49
CF
See coupling facility 52
CFRM couple data set 52, 66
CFRM policy 52
CHKTHRES common routine 99
clone ID
Automatic Restart Manager 157
code matching 84
command processing 84
commands
INGCF DRAIN 53
INGCF ENABLE 54
common automation items
defining 80
common global variables 99, 141
common routines 95
configuration file
sample for inetd 44
connector
active 52
failed persistent 52
continuous availability of Couple Data Sets
enabling 66
copying
SA OS/390 command server to the UNIX file system 30
couple data set
alternate, specifying 66
CFRM 66
SYSPLEX 66
couple data set (CDS) 49
alternate CDS 49
recovery 49
policy 49
primary CDS 49
Couple Data Sets
enabling continuous availability of 66
coupling facilities
managing 52
coupling facility (CF) 52
draining 53
enabling 54
CPCB (command processor control block) 177
customization
HealthChecker 78
of alternate CDS recovery 50
of auxiliary storage shortage 58
of hung command recovery 57
of IXC102A message automation 58
of SYSIEFSD resource recovery 57
of system log recovery 56
customization *(continued)*
of system logger recovery 51
of WTO(R) buffer shortage recovery 56
customization dialog exits 137
invocation 140
customization dialogs
defining OS/390 UNIX resources with 33
defining the inetd with 45
defininjng OS/390 UNIX resources with 34
customize automation
for processor operations 100
for system operations 98
customizing
SDF 157

D

debugging
OS/390 UNIX monitoring and command execution 44
defining
application type IMAGE 71
common automation items 80
IMAGE application type 71
logical partitions 64
OS/390 UNIX resources 33
OS/390 UNIX resources with customization dialogs 34
processor 63, 71
SDF in automation control file 172
started task job name 80
SYSPLEX policy item 66
system 65
temporary data set HLQ 80
UNIX segments (OMVS) 31
USS automation setup 34
defining IEADMC*xx* symbols
for long running enqueues 76
defining resources
for long running enqueues 75
defining the handling of jobs
for auxiliary storage shortage recovery 77
defining the local page data set
for auxiliary storage shortage recovery 77
directory extent 51
DISPASST 108
documents, licensed xvii
DSICMD member 107
DSIPARM
description 178
DSIPARM data set 107

E

element names
in Automatic Restart Manager 157
enabling
continuous availability of Couple Data Sets 66
ehanced Parallel Sysplex automation 49

enabling *(continued)*
system log failure recovery 67
system removal 70
WTOR(R) buffer shortage recovery 69
enhancement
for OS/390 UNIX System Services 29
ENQs
See enqueues 56
enqueues
long-running, handling 56
long-running, recovery customization 57
enqueues, long running
automating 74
environmental setup exits 132
error codes 99
EVJRSMON 93
exits 136
AOFEXC00 129
AOFEXC01 129
AOFEXC02 130
AOFEXC03 131
AOFEXC04 132
AOFEXDEF 33, 126
AOFEXI01 127
AOFEXI02 127
AOFEXI03 127
AOFEXI04 127
AOFEXINT 127, 137
AOFEXSTA 128
BUILDF processing 137
CONVERT processing 139
COPY processing 138
customization dialog exits 137
DELETE processing 139
environmental setup exits 132
flag exits 133
INGEAXIT 133
INGEX01 137
INGEX02 137
INGEX03 138
INGEX04 138
INGEX05 139
INGEX06 139
INGEX07 139
INGEX08 139
pseudo-exits 136
sample automation flag exits 136
static exits 126
status change commands 129
subsystem up at initialization commands 136
testing 137
EXPLAIN 142, 150
extended automation flags 15
extending
automation 95
external common global variables 141
EXTSTART status 157

F

failed persistent connector 52
failed system
isolation of 57
file manager commands 99
file monitoring
of OS/390 UNIX automated resources 40
file-system monitoring
of OS/390 UNIX automated resources 40
FINAL TERMINATION messages 18
flag exits 133

G

generic
automation 17, 149
routines 95
generic automation procedures 102
generic routines 95
global automation flag 16
graphic workstation description 183

H

HealthChecker
customization 78
hung command recovery 57
customization of 57
HWMCA
description 179

I

IDENT 112
IEADMC*xx* symbols, defining
for long running enqueues 76
IGNORE WTOR priority 22
IMAGE application type
defining 71
important considerations
processor operations 81
IMPORTANT WTOR priority 22
INCLUDE statement 171
inetd 44
defining with customization dialogs 45
sample configuration file 44
INGAUTO_INTERVAL 145, 153
INGCF command 53
DRAIN 53
ENABLE 54
PATH 55
STRUCTURE 55
INGDLG 140
INGEAXIT exit 133
INGEIO 180
INGEX01 137
INGEX02 137
INGEX03 138
INGEX04 138
INGEX05 139
INGEX06 139
INGEX07 139
INGEX08 139
INGGROUP_WAIT 145
INGMSG00 3
INGMSG01 3
INGPLEX command
CDS 50
INGREQ_ORIGINATOR 145, 154
INGREQ_OVERRIDE 145, 154
INGREQ_PRECHECK 145, 154
INGREQ_PRI 145, 154
INGREQ_REMOVE 146, 154
INGREQ_RESTART 146, 154
INGREQ_SCOPE 146, 154
INGREQ_TIMEOUT 146, 154
INGREQ_TYPE 146, 154
INGREQ_VERIFY 146, 155
INGREQ_WAIT 146
INGSCHED_WAIT 146
INGSET_VERIFY 146, 155
INGUSS 32, 40
JOBNAME parameter 33
INGVOTE_EXCLUDE 146
INGVOTE_STATUS 147
INGVOTE_VERIFY 147, 155
initialization processing
AOFSERXINT 144, 152
installing
OS/390 UNIX automation enhancements 30
installing automation procedures 107
invoking
internal monitoring of USS instances 37
UNIX monitoring routine 37
IODF 55
ISQEXEC command 87, 101
ISQMTSYS 93
ISQOVRD 88
ISQOVRD command 102
ISQXLOC command 101
ISQXMON command 87
ISQXUNL command 101
ISSUECMD 15, 84
ISSUEREP 15, 84
IWTOR 22
IXC102A message
automation of 57
customization of automation of 58
IXC102A message automation 118
IXC102A, message
automating 72
IXC402D message
automation of 57
IXC402D, message
automating 72

J

job handling, defining
for auxiliary storage shortage recovery 77
job/ASID definitions, making
for long running enqueues 75

L

licensed documents xvii
Linux console connection to NetView 122
local page data set, defining
for auxiliary storage shortage recovery 77

log stream 50
log stream data set 51
logical partitions
  defining 64
LOGR couple data set 51
long running enqueues
  automating 74
  defining IEADMC*xx* symbols 76
  defining resources 75
  making job/ASID definitions 75
long-running enqueues
  handling 56
LookAt message retrieval tool xvii

M

major resources 15, 135
making job/ASID definitions
  for long running enqueues 75
master automation tables 2
  multiple 13
message
  ISQ900I 86
  ISQ901I 86
  types 1
message automation 118
message automation for IXC102A 118
message automation for Linux console messages 122
message automation for processor operations resources 115
message presentation 4
message processing
  user specific 85
message processing facility list
  adding application messages 23
message retrieval tool, LookAt xvii
message testing 89
messages
  assigning to automation environment 90
  automating response to 83
  classifications 1
  defining as minor resources 15
  forwarding 86
  status transition 17
  suppressing 91
  testing 87, 89
messages, entries in MPF list 23
minor resources
  and INGAUTO 15
  and task globals 135
  AOFMINORCHK 151
  defining message and status 15
  resource name 135
monitor routine 93
  writing own 93
monitoring
  automation with interactive assist mode 109
  enable an application 27
monitoring applications 93
monitoring routine
  AOFUXMON 38
MPF list 27
  adding an entry 90
  adding application messages 23
MVS Automatic Restart Manager
  global variables 157
MVSESA.RELOAD.ACTION minor resource 136
MVSESA.RELOAD.CONFIRM flag 136
MVSESA.RELOAD.CONFIRM minor resource 136

N

NetView
  customizing for OS/390 UNIX automation enhancement 33
  generic automation table entries 17
  testing and debugging facilities 111
NetView automation table
  adding entry calling ACTIVMSG 23
  adding entry calling ACTIVMSGUP=YES 23
  adding entry calling TERMMSG 24
  adding entry calling TERMMSG FINAL=YES 24
  adding SDF entries 25
  ISQEXEC 87, 101
  ISQOVRD 102
  ISQXLOC 101
  ISQXMON 87
  ISQXUNL 101
  merging entries 90
  performance consideration 85
  production 89
  reloading tables 27
  sample entry 87
NetView message automation table 83
  adding an entry 85
  generic entries 17
NMC focal point system 178
NMC workstation 27
NONSNA 27
NORMAL WTOR priority 22
notifications 98
NWTOR 22

O

OMVS segment
  creating by submitting a job 32
  defining 31
  using with non-root UID 31
    command execution (INGUSS) 32
    monitoring 31
    setup restrictions 32
  using with root UID 31
operator xiii
operator cascades 8
OS/390 UNIX automated resources
  defining with customization dialogs 39
  file or file-system monitoring 40
  process monitoring 39
  TCP port monitoring 39
OS/390 UNIX automation enhancement
  BPX.JOBNAME 33
  copying the SA OS/390 command server 30
OS/390 UNIX automation enhancement *(continued)*
  customization of OS/390 UNIX resources 33
  customizing NetView 33
  defining the UNIX segments for 31
  hints and tips 46
  installing and setting up 30
  restarting NetView 33
OS/390 UNIX command execution
  debugging 44
OS/390 UNIX monitoring
  debugging 44
OS/390 UNIX resource
  defining with customization dialogs 34
OS/390 UNIX resources
  command examples 42
  defining 33
  file stop command example 43
  INGUSS command 40
  initial user environment 41
  obtaining unique jobnames 42
  process start command example 42
  process stop command example 43
  softlink command example 42
  start and stop definitions 40
OS/390 UNIX System Services
  enhancement 29
  resources, automating 29
outstanding reply processing 21

P

persistent connection 52
persistent structure 52
policy 49
  CFRM 52
preference list 52
PRI WTOR type 22
primary CDS 49
process monitoring
  of OS/390 UNIX automated resources 39
processor
  defining 63, 71
PROCESSOR INFO policy item
  using 64
processor operations
  dedicated personal computer 185
  important considerations 81
processor operations command messages 88
processor operations commands 103
processor operations controlled resources
  automating 115
processor operations resource 115
  message automation 115
product automation 185
programming recommendations 112
proxy resource 115
proxy resources
  automation for 116
pseudo-exits 136

R

RACF facility class
  BPX.JOBNAME 33
rebuild 52
  system-managed 52
  user-managed 52
recovery
  alternate CDS 49
    customization 50
  auxiliary storage shortage 58
  auxiliary storage shortage, automating 76
  handling long-running enqueues 56
  hung command 57
  long-running enqueues
    customization 57
  SYSIEFSD resource 56
  system log 55
    customization 56
  system log failure, enabling 67
  system logger
    customization 51
    directory shortage 51
    VSAM share options 51
  WTO(R) buffer shortage 56
    customization 56
  WTOR(R) buffer shortage, enabling 69
reload action exit 136
reload permission exit 136
RELOAD.ACTION flag 136
RELOAD.CONFIRM flag 136
reloading NetView automation table 27
reply processing 84
  outstanding 21
resident clists
  adding AOFUXMON and AOFRSUSS 33
resolving
  pending I/Os for systems being removed from the sysplex
    customization of 58
  system log failure 55
  WTO(R) buffer shortages 56
resources, defining
  for long running enqueues 75
REXX PARSE 112
REXX trace type 110
routines
  common 95
  generic 95
RWTOR 22

S

SA OS/390
  commands ISQXIPM and ISQCMMT 100
SA OS/390 command server
  copying to the UNIX file system 30
sample
  automation tables 1
SDF
  and specific problems 164
  components 166
  customizing 157
SDF *(continued)*
  customizing initialization parameters 171
  defining hierarchy 168
  defining in automation control file 172
  defining in customization dialog 172
  defining panels 169
  definition process 167
  for multiple systems 165
  how it works 157
  panels
    definition 164, 168
    types 157
  starting and stopping 166
  status descriptors 158
  tree structures 159
SDF entries 25
SDF panels 26
SDF tree structure 25
SEC WTOR type 22
serialize command processing 100
SETASST 108
setting
  AOFUSSWAIT 33
setting up
  OS/390 UNIX automation enhancements 30
SFM
  See Sysplex Failure Management 57
start definition
  OS/390 UNIX resources 40
started task job name
  defining 80
status
  defining as minor resources 15
status change commands 129
status descriptors 160
  chaining to status components 161
  propagating 163
status file 175
status information 98
status transition messages 17
stop definition
  OS/390 UNIX resources 40
structure 52
  allocation 52
  deallocation 52
  duplexing 53
  persistent 52
  preference list 52
  rebuild 52
    system-managed 52
    user-managed 52
SUBSAPPL 112
SUBSJOB 112
SUBSTYPE 112
subsystem
  adding to automation 21
  up at initialization commands 136
suppressing messages 91
SYSIEFSD resource recovery 56
  customization of 57
SYSLOG message automation 55
SYSPLEX couple data set 66
Sysplex Failure Management (SFM) 57
SYSPLEX policy item
  defining 66
system
  defining 65
system log 55
system log failure
  recovery, enabling 67
system log recovery
  customization of 56
system logger
  directory extent 51
  log stream 50
  log stream data set 51
    VSAM share options 51
  LOGR couple data set 51
system logger recovery
  customization of 51
system operations control files 91
  automation control file 22
  automation manager configuration file 22
system programmer xiii
system removal 57
  enabling 70
system-managed rebuild 52

T

task global variables 99
TCP port monitoring
  of OS/390 UNIX automated resources 39
temporary data set HLQ
  defining 80
TERMMSG 18, 24
testing
  messages 89
testing and debugging automation procedures 107
testing and debugging facilities 111
testing exits 137
trapping UNIX syslogd messages 46

U

UCB (unit control block) 190
UNIX file system
  copying the SA OS/390 command server to 30
UNIX internet daemon
  See inetd 44
UNIX monitoring routine
  exit values 37
  invoking 37
UNIX segments (OMVS)
  defining 31
UNIX syslogd messages
  trapping 46
UNIX System Services resources
  automating 29
UNUSUAL WTOR priority 22
usage
  of &SUBSUSSJOB and &SUBSASID 41
user exits 125
  static exits 126

user groups
  automation programmer xiii
  operator xiii
  system programmer xiii
user specific message processing 85
user-managed rebuild 52
using
  PROCESSOR INFO policy item 64
USS automation setup
  defining 34
USS resources
  automating 29
  infrastructure overview 29
UWTOR 22

V

VTAM
  and assist mode 108

W

WAITTIME 147
WTO(R) buffer 56
WTO(R) buffer shortage recovery
  customization of 56
WTOR
  priority 21
  type 21
WTOR(R) buffer shortage
  recovery, enabling 69

X

XDOMTIME 147

Readers' Comments — We'd Like to Hear from You

System Automation for OS/390
Customizing and Programming
Version 2 Release 2

Publication No. SC33-7035-04

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

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.

Name

Address

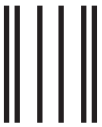
Company or Organization

Phone No.

Readers' Comments — We'd Like to Hear from You
SC33-7035-04

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

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

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Program Number: 5645-006

Printed in U.S.A.

SC33-7035-04

Spine information:

System Automation for OS/390 **Customizing and Programming** Version 2 Release 2 SC33-7035-04