

System Automation for OS/390

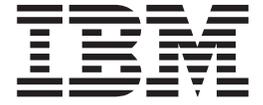


# Customizing and Programming

*Version 2 Release 1*



System Automation for OS/390



# Customizing and Programming

*Version 2 Release 1*

**Note!**

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

**Second Edition (December 2001)**

This edition applies to System Automation for OS/390 (Program Number 5645-006) Version 2, Release 1, 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

**Internet**

Visit our homepage at <http://www.ibm.com/servers/eserver/zseries/software/sa/>

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

© Copyright International Business Machines Corporation 1996, 2001. All rights reserved.

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

# Contents

<b>Figures</b> . . . . .	<b>vii</b>
--------------------------	------------

<b>Tables</b> . . . . .	<b>ix</b>
-------------------------	-----------

<b>Notices</b> . . . . .	<b>xi</b>
--------------------------	-----------

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

<b>About This Book</b> . . . . .	<b>xiii</b>
----------------------------------	-------------

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 licensed books on the Web . . . . .	xviii

<b>Summary of Changes</b> . . . . .	<b>xix</b>
-------------------------------------	------------

What's New in This Book for SA OS/390 2.1 . . . . .	xix
Notes on Terminology . . . . .	xix

<b>Chapter 1. Introduction</b> . . . . .	<b>1</b>
--	----------

<b>Chapter 2. A Guide to SA OS/390</b>	
--	--

<b>Automation Table Samples</b> . . . . .	<b>3</b>
---	----------

SA OS/390 Message Types . . . . .	3
Automation Table Structure . . . . .	3
Master Automation Tables . . . . .	4
AOFMSGSY . . . . .	4
SA OS/390 Message Presentation . . . . .	5
Operator Cascades . . . . .	9
Automation Table Fragments . . . . .	15
AOFMAPPCC . . . . .	15
AOFMGMFH. . . . .	16
AOFMIO . . . . .	16
AOFMJES2 . . . . .	16
AOFMJES3 . . . . .	16
AOFMMVS . . . . .	17
AOFMNETV . . . . .	17
AOFMPOPS . . . . .	17
AOFMRMF . . . . .	18
AOFMRODM. . . . .	18
AOFMSGSY . . . . .	18
AOFMSGGW. . . . .	18
AOFMSGST . . . . .	19
AOFMSGTS . . . . .	19
AOFMSGPM . . . . .	19
AOFMSGPA . . . . .	19
AOFMSGMS . . . . .	20
AOFMSGHI . . . . .	20
AOFMSGBL . . . . .	20
AOFMSGGN . . . . .	20
AOFMSGNM. . . . .	21

AOFMSGXC . . . . .	21
AOFMTSO . . . . .	21
AOFMVLLA . . . . .	22
AOFMVTAM. . . . .	22
INGMTEC. . . . .	22
Integrating Automation Tables . . . . .	22
Multiple Master Automation Tables . . . . .	23
Black Box . . . . .	23
Insert into AOFMSG01 . . . . .	24
Insert into Fragments . . . . .	24
Include SA OS/390 Fragments . . . . .	24

<b>Chapter 3. How to Automate Your Resources</b> . . . . .	<b>25</b>
--	-----------

Using Automation Flags . . . . .	25
Example . . . . .	25
When SA OS/390 Checks Automation Flags . . . . .	26
The Automation Manager Global Automation Flag . . . . .	26
Generic Automation Table Entries . . . . .	27
Status Transition Messages . . . . .	27
Abnormal End (Abend) Messages . . . . .	28
Other Messages . . . . .	29

<b>Chapter 4. How to Add a New Application to Automation</b> . . . . .	<b>31</b>
--	-----------

Step 1: Defining an Application Policy Object . . . . .	31
Step 2: Defining Outstanding Reply Processing . . . . .	31
Step 3: Building New System Operations Control Files . . . . .	32
Step 4: Coding Entries for Application Messages in the MPF List . . . . .	33
Step 5: Adding an Entry Calling ACTIVMSG to the NetView Message Automation Table . . . . .	33
Adding an Entry Calling ACTIVMSG UP=YES to the NetView Message Automation Table. . . . .	33
Step 6: Adding an Entry Calling TERMMSG to the NetView Message Automation Table . . . . .	34
Adding an Entry Calling TERMMSG FINAL=YES to the NetView Message Automation Table . . . . .	34
Step 7: Adding SDF Entries for the Subsystem. . . . .	35
Updating SDF Tree Structure . . . . .	35
Updating SDF Panels . . . . .	36
Dynamically Loading SDF Tree Structure and Panels . . . . .	36
Step 8: Enable the Application for Monitoring . . . . .	37
Step 9: Reloading Tables . . . . .	37

<b>Chapter 5. How to Add a Message to Automation.</b> . . . . .	<b>39</b>
---	-----------

Using Multiple NetView Message Automation Tables . . . . .	39
Step 1: Automating the Response to the Message . . . . .	39
Responding with a Command . . . . .	40

Responding with a Reply . . . . .	40
Responding with Code Matching . . . . .	40
Creating User Specific Message Processing . . . . .	41
Step 2: Adding an Entry in the NetView Message Automation Table . . . . .	41
Messages Issued by an Application or MVS Component . . . . .	41
Messages Issued by a Processor Operations Target System . . . . .	43
Sample NetView Automation Table Entries . . . . .	44
Message ISQ055I . . . . .	44
Processor Operations Command Messages . . . . .	44
Testing Messages . . . . .	46
Step 3: Assigning the Message to the Automation Environment . . . . .	47
Adding a NetView Automation Operator ID . . . . .	47
Step 4: Building the New Automation Definitions . . . . .	47
Step 5: Loading the Changed Automation Environment . . . . .	48

**Chapter 6. How to Write Your own Monitor Routines . . . . . 49**

**Chapter 7. How to Create Automation Procedures . . . . . 51**

Programming Additional SA OS/390 Automation Procedures . . . . .	51
How Generic Routines and Common Routines Are Used in Automation Procedures . . . . .	51
How Automation Procedures Are Called . . . . .	52
How Automation Procedures Are Structured . . . . .	52
Performing Initialization Processing . . . . .	53
Determining Whether Automation Is Allowed . . . . .	54
Performing Automation Processing . . . . .	54
How to Make Your Automation Procedures Generic . . . . .	59
Processor Operations Commands . . . . .	60
Developing Messages for Your Automation Procedures . . . . .	60
Example AOCMSG Call . . . . .	61
Adding NetView Message Automation Table Entries . . . . .	61
Example Automation Procedure . . . . .	62
Notes on the Automation Procedure Example . . . . .	63
Installing Your Automation Procedures . . . . .	64
Testing and Debugging Automation Procedures . . . . .	64
The Assist Mode Facility . . . . .	64
Using Assist Mode to Test Automation Procedures . . . . .	65
Using AOCTRACE to Trace Automation Procedure Processing . . . . .	67
NetView Testing and Debugging Facilities . . . . .	69
Where to Find More Testing Information . . . . .	69
Coding Your Own Information in the Automation Status File . . . . .	69
Programming Recommendations . . . . .	70
Global Variable Names . . . . .	71

**Chapter 8. How to Automate Processor Operations Controlled Resources . . . 73**

Automating Processor Operations Resources of OS/390 Target Systems Using Proxy Definitions . . . . .	73
The Concept . . . . .	73
Customizing Automation for Proxy Resources . . . . .	74
Preparing Message Automation . . . . .	76
Message Automation for IXC102A . . . . .	77
Loss of Focal Point Authorization of a Support Element . . . . .	77
How to Customize IXC102A Automation . . . . .	77
Automating Linux Console Messages . . . . .	82
The Linux Console Connection to NetView . . . . .	82
Linux Console Automation with Mixed Case Character Data . . . . .	82
Security Considerations . . . . .	82
Restrictions and Limitations . . . . .	82

**Chapter 9. SA OS/390 User Exits . . . 83**

Static Exits . . . . .	84
AOFEXDEF . . . . .	84
AOFEXI01 . . . . .	85
AOFEXI02 . . . . .	85
AOFEXINT . . . . .	85
AOFEXSTA . . . . .	85
AOFEXC00 . . . . .	87
AOFEXC01 . . . . .	87
Environmental Setup Exits . . . . .	88
Flag Exits . . . . .	89
Parameters . . . . .	91
Return Codes . . . . .	91
Pseudo-Exits . . . . .	92
Automation Control File Reload Permission Exit . . . . .	92
Automation Control File Reload Action Exit . . . . .	92
Subsystem Up at Initialization Commands . . . . .	93
Testing Exits . . . . .	93
Customization Dialog Exits . . . . .	93
User Exits for BUILD Processing . . . . .	93
User Exits for COPY Processing . . . . .	94
User Exits for DELETE Processing . . . . .	94
User Exits for CONVERT Processing . . . . .	95
Invocation of Customization Dialog Exits . . . . .	95

**Chapter 10. Maintaining the Processor Operations PC . . . . . 97**

Editing the Profile Data Set . . . . .	97
Keeping PS/2 Configuration Data Synchronized . . . . .	98
PS/2 Configuration Data Cross-Reference . . . . .	99

**Appendix A. Table of External Global Variables . . . . . 107**

**Appendix B. Global Variables to Enable Advanced Automation . . . . 113**

**Appendix C. Customizing the Status Display Facility (SDF) . . . . . 121**

Overview of Status Display Facility . . . . .	121
How SDF Works . . . . .	121
Types of SDF Panels . . . . .	121
Status Descriptors . . . . .	122

SDF Tree Structures . . . . .	123
How Status Descriptors Affect SDF . . . . .	125
How SDF Helps Operations to Focus on Specific Problems . . . . .	128
How SDF Panels Are Defined . . . . .	128
Dynamically Loading Tree Structure and Panel Definition Members . . . . .	129
Using SDF for Multiple Systems . . . . .	129
SDF Components . . . . .	130
How the SDF Task Is Started and Stopped . . . . .	131
SDF Definition . . . . .	131
Summary of SDF Definition Process . . . . .	131

Step 1: Defining SDF Hierarchy . . . . .	132
Step 2: Defining SDF Panels . . . . .	133
Step 3: Customizing SDF Initialization Parameters . . . . .	136
Step 4: Defining SDF in the Customization Dialog. . . . .	136

<b>Glossary . . . . .</b>	<b>137</b>
---------------------------	------------

<b>Index . . . . .</b>	<b>157</b>
------------------------	------------



---

## Figures

1. Example of a WTORS Entry . . . . .	31	10. Flag Automation Specification Panel (1)	81
2. Automation Procedures for System Operations	53	11. SA OS/390 Exit Sequence during SA OS/390 Initialization . . . . .	84
3. Automation Procedures for Processor Operations . . . . .	53	12. Example SDF Panels . . . . .	122
4. Skeleton of an Automation Procedure . . . . .	60	13. Example SDF Tree Structure . . . . .	124
5. SDF Detail Status Display Panel with Assist Mode . . . . .	67	14. Status Descriptors Chained to Status Components . . . . .	126
6. Message Processing Panel. . . . .	78	15. Example Tree Structure Definition. . . . .	132
7. CMD Processing Panel. . . . .	79	16. Example SDF Panel . . . . .	134
8. Code Processing Panel . . . . .	79	17. Example Panel Definition Entry . . . . .	134
9. Minor Resource Selection Panel . . . . .	80		



---

## Tables

1.	System Automation for OS/390 Library	xiv	6.	Externalized Common Global Variables	107
2.	Related Products Books . . . . .	xv	7.	Global Variables to Enable Advanced Automation (CGLOBALS) . . . . .	113
3.	Automation Flags: Typical Uses in SA OS/390	26	8.	SDF Components . . . . .	130
4.	Synchronizing PS/2 Configuration Data	99	9.	Panel Definition Entry Description . . . . .	134
5.	PS/2 Data Cross-Reference . . . . .	100			



---

## 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
BookMaster	CICS
DB2	DFSMS/MVS
ES/9000	ESCON
IBM	IMS
Library Reader	Micro Channel
MQSeries	Multiprise
MVS	MVS/ESA
MVS/XA	Netfinity
Netfinity Manager	OpenEdition
OS/2	OS/390
Parallel Sysplex	RACF
RMF	Sysplex Timer
System/370	System/390
Tivoli	Tivoli Enterprise Console
VM/ESA	VSE/ESA
VTAM	3090

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. You can customize the automation performed by SA OS/390 by, for example:

- Specifying new applications that should be automated by SA OS/390
- Adding new messages of automated applications and specifying how SA OS/390 should react if these messages occur
- Writing your own automation procedures that are executed under specified conditions

---

## 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
<i>System Automation for OS/390 General Information</i>	GC33-7036
<i>System Automation for OS/390 Licensed Program Specifications</i>	SC33-7037
<i>System Automation for OS/390 Planning and Installation</i>	SC33-7038
<i>System Automation for OS/390 Customizing and Programming</i>	SC33-7035
<i>System Automation for OS/390 Defining Automation Policy</i>	SC33-7039
<i>System Automation for OS/390 User's Guide</i>	SC33-7040
<i>System Automation for OS/390 Messages and Codes</i>	SC33-7041
<i>System Automation for OS/390 Operator's Commands</i>	SC33-7042
<i>System Automation for OS/390 Programmer's Reference</i>	SC33-7043
<i>System Automation for OS/390 CICS Automation Programmer's Reference and Operator's Guide</i>	SC33-7044
<i>System Automation for OS/390 IMS Automation Programmer's Reference and Operator's Guide</i>	SC33-7045
<i>System Automation for OS/390 OPC Automation Programmer's Reference and Operator's Guide</i>	SC23-7046

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

IBM Online Library OS/390 Collection (SK2T-6700)

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

This softcopy collection includes the IBM Library Reader, a program that enables you to view online documentation.

**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
<i>ISPF User's Guide</i>	SC34-4484
<i>ISPF Dialog Management Guide and Reference</i>	SC34-4266
<i>MVS/ESA MVS Configuration Program Guide and Reference</i>	GC28-1817
<i>MVS/ESA Planning: Dynamic I/O Configuration</i>	GC28-1674
<i>MVS/ESA Support for the Enterprise Systems Connection</i>	GC28-1140
<i>MVS/ESA Planning: APPC Management</i>	GC28-1110
<i>MVS/ESA Application Development Macro Reference</i>	GC28-1822
<i>MVS/ESA SP V5 System Commands</i>	GC28-1442

Table 2. Related Products Books (continued)

<b>Title</b>	<b>Order Number</b>
<i>MVS/ESA SPL Application Development Macro Reference</i>	GC28-1857
<i>OS/390 Hardware Configuration Definition: User's Guide</i>	SC28-1848
<i>OS/390 Information Roadmap</i>	GC28-1727
<i>OS/390 Information Transformation</i>	GC28-1985
<i>OS/390 Introduction and Release Guide</i>	GC28-1725
<i>OS/390 JES Commands Summary</i>	GX22-0041
<i>OS/390 Licensed Program Specifications</i>	GC28-1728
<i>OS/390 Printing Softcopy Books</i>	S544-5354
<i>OS/390 Starting Up a Sysplex</i>	GC28-1779
<i>OS/390 Up and Running!</i>	GC28-1726
<i>Planning for the 9032 Model 3 and 9033 Enterprise Systems Connection Director</i>	SA26-6100
<i>Resource Access Control Facility (RACF) Command Language Reference</i>	SC28-0733
<i>S/390 MVS Sysplex Overview – An Introduction to Data Sharing and Parallelism</i>	GC23-1208
<i>S/390 MVS Sysplex Systems Management</i>	GC23-1209
<i>S/390 Sysplex Hardware and Software Migration</i>	GC23-1210
<i>S/390 MVS Sysplex Application Migration</i>	GC23-1211
<i>S/390 Managing Your Processors</i>	GC38-0452
<i>Tivoli/Enterprise Console User's Guide Volume I</i>	GC31-8334
<i>Tivoli/Enterprise Console User's Guide Volume II</i>	GC31-8335
<i>Tivoli/Enterprise Console Event Integration Facility Guide</i>	GC31-8337
<i>Tivoli for OS/390 NetView V1R3 Administration Reference</i>	SC31-8222
<i>Tivoli for OS/390 NetView V1R3 Application Programmer's Guide</i>	SC31-8223
<i>Tivoli for OS/390 NetView V1R3 APPN Topology and Accounting Agent Guide</i>	SC31-8224
<i>Tivoli for OS/390 NetView V1R3 Automation Guide</i>	SC31-8225
<i>Tivoli for OS/390 NetView V1R3 AON Customization Guide</i>	SC31-8662
<i>Tivoli for OS/390 NetView V1R3 AON User's Guide</i>	GC31-8661
<i>Tivoli for OS/390 NetView V1R3 Bridge Implementation</i>	SC31-8238
<i>Tivoli for OS/390 NetView V1R3 Command Reference Vol. 1</i>	SC31-8227
<i>Tivoli for OS/390 NetView V1R3 Command Reference Vol. 2</i>	SC31-8227
<i>Tivoli for OS/390 NetView V1R3 Customization Guide</i>	SC31-8228
<i>Tivoli for OS/390 NetView V1R3 Customization: Using Assembler</i>	SC31-8229
<i>Tivoli for OS/390 NetView V1R3 Customization: Using Pipes</i>	SC31-8248
<i>Tivoli for OS/390 NetView V1R3 Customization: Using PL/I and C</i>	SC31-8230
<i>Tivoli for OS/390 NetView V1R3 Customization: Using REXX and the NetView Command List Language</i>	SC31-8231
<i>Tivoli for OS/390 NetView V1R3 Data Model Reference</i>	SC31-8232
<i>Tivoli for OS/390 NetView V1R3 Installation and Administration Guide</i>	SC31-8236
<i>Tivoli for OS/390 NetView V1R3 Messages and Codes</i>	SC31-8237

Table 2. Related Products Books (continued)

Title	Order Number
<i>Tivoli for OS/390 NetView V1R3 MultiSystem Manager User's Guide</i>	SC31-8607
<i>Tivoli for OS/390 NetView V1R3 NetView Graphic Monitor Facility User's Guide</i>	GC31-8234
<i>Tivoli for OS/390 NetView V1R3 NetView Management Console User's Guide</i>	GC31-8665
<i>Tivoli for OS/390 NetView V1R3 User's Guide</i>	SC31-8241
<i>Tivoli for OS/390 NetView V1R3 Planning Guide</i>	GC31-8226
<i>Tivoli for OS/390 NetView V1R3 RODM and GMFHS Programmer's Guide</i>	SC31-8233
<i>Tivoli for OS/390 NetView V1R3 Security Reference</i>	SC31-8606
<i>Tivoli for OS/390 NetView V1R3 SNA Topology Manager and APPN Accounting Manager Implementation Guide</i>	SC31-8239
<i>Tivoli Management Platform Reference Guide</i>	GC31-8324
<i>TSO/E REXX/MVS User's Guide</i>	SC28-1882
<i>TSO/E REXX/MVS Reference</i>	SC28-1883
<i>VM/XA SP GCS Command and Macro Reference</i>	SC23-0433
<i>VSE/SP Unattended Node Support</i>	SC33-6412
<i>VTAM Messages and Codes</i>	SC31-6493
<i>VTAM Network Implementation Guide</i>	SC31-6404
<i>VTAM Network Implementation Guide</i>	SC31-6434

## Using LookAt to look up message explanations

LookAt is an online facility that allows you to look up explanations for z/OS messages, system abends, and some 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/servers/eserver/zseries/zos/bkserv/lookat/lookat.html>

or from anywhere in z/OS where you can access a TSO command line (for example, TSO prompt, ISPF, z/OS UNIX System Services running OMVS).

To find a message explanation on the Internet, go to the LookAt Web site and simply enter the message identifier (for example, IAT1836 or IAT\*). You can select a specific release to narrow your search. You can also download code from the *z/OS Collection*, SK3T-4269 and the LookAt Web site so you can access LookAt from a PalmPilot (Palm VIIx suggested).

To use LookAt as a TSO command, you must have LookAt installed on your host system. You can obtain the LookAt code for TSO from a disk on your *z/OS Collection*, SK3T-4269 or from the LookAt Web site. To obtain the code from the LookAt Web site, do the following:

1. Go to <http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookat.html>.
2. Click the **News** button.
3. Scroll to **Download LookAt Code for TSO and VM**.

4. Click the ftp link, which will take you to a list of operating systems. Select the appropriate operating system. Then select the appropriate release.
5. Find the **lookat.me** file and follow its detailed instructions.

To find a message explanation from a TSO command line, simply enter: **lookat message-id**. LookAt will display the message explanation for the message requested.

**Note:** Some messages have information in more than one book. For example, IEC192I has routing and descriptor codes listed in *z/OS MVS Routing and Descriptor Codes*. For such messages, LookAt prompts you to choose which book to open.

## Accessing licensed books on the Web

OS/390 licensed documentation in PDF format is available on the Internet at the IBM Resource Link Web site at:

<http://www.ibm.com/servers/resourceLink>

Licensed books are available only to customers with a OS/390 license. Access to these books requires an IBM Resource Link Web userid and password, and a key code. With your OS/390 order you received a memo that includes this key code.

To obtain your IBM Resource Link Web userid and password log on to:

<http://www.ibm.com/servers/resourceLink>

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

1. Log on to Resource Link using your Resource Link userid and password.
2. Click on **User Profiles** located on the left-hand navigation bar.
3. Click on **Access Profile**.
4. Click on **Request Access to Licensed books**.
5. Supply your key code where requested and click on the **Submit** button.

If you supplied the correct key code you will receive confirmation that your request is being processed. After your request is processed you will receive an e-mail confirmation.

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

To access the licensed books:

1. Log on to Resource Link using your Resource Link userid and password.
2. Click on **Library**.
3. Click on **zSeries**.
4. Click on **Software**.
5. Click on **OS/390**.
6. Access the licensed book by selecting the appropriate element.

---

## Summary of Changes

---

### What's New in This Book for SA OS/390 2.1

This book is a new book for the SA OS/390 2.1 library. It is the result of a task-oriented restructuring of the SA OS/390 library, and as such it does not, in the first place, contain new information about new SA OS/390 2.1 functions. Rather, it contains information required for adapting your completed standard SA OS/390 installation as described in *System Automation for OS/390 Planning and Installation* to your environment. Information required to achieve this has been previously presented in *System Automation for OS/390 Defining Automation Policy* and *System Automation for OS/390 Planning and Installation* and has now been updated and consolidated in this book.

“Chapter 8. How to Automate Processor Operations Controlled Resources” on page 73 contains information about new SA OS/390 2.1 functionality.

---

### Notes on Terminology

#### Processor Operations PC

Throughout this publication the term *processor operations PC* (*ProcOps PC*) is used to refer to the personal computer (PC) that is connected to one or more target systems and which contains the processor operations distributed feature code. In the worksheets and SA OS/390 panels, the terms “PC”, “PS/2” or “ProcOps PS/2” are used to refer to this workstation. The workstation may be a Micro Channel<sup>®</sup> Architecture (MCA) bus personal computer, a PCI bus personal computer, or an IBM-compatible ISA bus personal computer.

#### MVS:

References in this book to “MVS” refer either to the MVS/ESA<sup>™</sup> product or to the MVS<sup>™</sup> element of OS/390.



---

## Chapter 1. Introduction

This book primarily contains information on how to customize your SA OS/390 product after a successful standard installation. For example, it contains information on how to add new applications to automation or how to write your own automation procedures.

However, you must read the information contained in “Chapter 2. A Guide to SA OS/390 Automation Table Samples” on page 3 to be able to start the automation offered by SA OS/390 successfully.



---

## Chapter 2. A Guide to SA OS/390 Automation Table Samples

SA OS/390 ships three loadable automation tables and many automation table fragments. This section describes them. You will find it useful to have a copy of *NetView for MVS Automation Implementation* on hand as it explains the rules for the coding of the NetView automation table. If something is not explained here, it will probably be explained in this NetView book or in *Tivoli NetView for OS/390 Automation Guide*.

**Note:** *NetView for MVS Automation Implementation* 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 rationale behind the table's design, the SA OS/390 message types are explained below.

#### **Type I - Information**

These messages are of an informational and non-urgent nature. SA OS/390 produces many of them.

#### **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 ships three loadable automation tables. These are known as *master tables*. Each master table includes some automation table statements and one or more %INCLUDE statements to pull in various fragments containing related automation table statements. There are three major classifications of the fragments:

## A Guide to SA OS/390 Automation Table Samples

### 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 that the synonyms be suitably customized to reflect your environment. Details of the customization steps required are given later.

### SA OS/390 Functional Definitions

These fragments (all named AOFMSGxx) contain automation table statements related to specific functions of SA OS/390. Although you should not need to change any of them, you may need to modify the master tables to exclude those fragments related to functions that you are not using. These are detailed later.

### 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). Details of these, and notes on adding further automation for the applications, are given later.

## Master Automation Tables

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

### AOFMSG00

This is a boots trap automation table. It contains a small number of entries that are required to get your SA OS/390 NetView initialized. It should be %INCLUDED into your initial automation table. When SA OS/390 has finished loading its automation control file data, an entry in this table triggers further initialization processing that includes the loading of the primary automation table specified in the automation control file.

This table makes use of the synonyms defined in AOFMSGSY.

### AOFMSG01

This is suitable for use as a primary automation table. Review the member and optionally put into comments the %INCLUDEs that you do not require and un-comment those that you require. See “Chapter 2. A Guide to SA OS/390 Automation Table Samples” on page 3 for information on integrating it with your automation table statements.

If you are running only one NetView on the workstation focal point system, un-comment the %INCLUDE for AOFMSGST in the AOFMSG01 member for this system.

### AOFMSG03

This is a table suitable for inclusion into your primary table on a NetView that has the SA OS/390 Satellite installed upon it.

---

## 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.

## A Guide to SA OS/390 Automation Table Samples

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
%AOFDEFACT%	<p>This synonym is used at the head of the AOFMSG01 master automation table. It sets the default action flags for all messages that are processed by the automation table. If a later trap in the table does not change the actions specified here, these defaults are used. Be careful if you change %AOFDEFACT% as it affects many messages.</p> <p><b>Default</b></p> <p>SYSLOG(N) NETLOG(Y) DISPLAY(Y)</p> <p>This ensures that messages processed by the automation table are written to the netlog, but not to the syslog, and displayed to the user if applicable.</p>
%AOFDOM%	<p>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.</p> <p><b>Default</b></p> <p>&amp;DOMAIN.</p> <p>This is a default domain name used in a number of the samples.</p>
%AOFSYS%	<p>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.</p> <p>If you are running in a PolyJes environment, where you have multiple system names on the same machine, you must define additional synonyms for the extra system names and modify the AOFMSGBL and AOFMJESx fragments to reflect this.</p> <p><b>Default</b></p> <p>&amp;SYSNAME.</p> <p>This is a default system name used in a number of the samples.</p>

### SA OS/390 Message Presentation

The presentation of SA OS/390 messages (prefixed with AOF and ING) under NetView is controlled by automation table fragment AOFMSGHI. 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

## A Guide to SA OS/390 Automation Table Samples

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. The synonym is used in fragment AOFMSGHI.

### 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. The synonym is used in fragment AOFMSGHI.

### 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. The synonym is used in fragment AOFMSGHI.

### 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. The synonym is used in fragment AOFMSGHI.

### Default

## A Guide to SA OS/390 Automation Table Samples

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. The synonym is used in fragment AOFMSGHI.

#### **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**

## A Guide to SA OS/390 Automation Table Samples

'AUT01' and 'AUT02'

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. The synonym is used in fragment AOFMSGHI.

#### **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. The synonym is used in fragment AOFMSGHI.

#### **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. The synonym is used in fragment AOFMSGHI.

#### **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**

## A Guide to SA OS/390 Automation Table Samples

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. The synonym is used in fragment AOFMSGHI.

### 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. The synonym is used in fragment AOFMSGHI.

### 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 AUTOBASE 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

## A Guide to SA OS/390 Automation Table Samples

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

## A Guide to SA OS/390 Automation Table Samples

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 AUTOBASE 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 AUTOBASE 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 AUTOBASE 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 AUTOBASE AUTO1

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

### **%AOFOPRECOPER%**

#### Usage

## A Guide to SA OS/390 Automation Table Samples

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 AUTOBASE 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 AUTOBASE 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 AUTOBASE 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 113) 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 jobname.

### **%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%.

## A Guide to SA OS/390 Automation Table Samples

### Default

```
AUTSYS AUTOBASE AUTO1
```

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

### **%AOFOPMSU%**

#### Usage

This cascade is used to route commands concerning MSUs. If you are not using the standard names for SA OS/390 autotasks you must change this synonym. It is mainly used on the Satellite SA OS/390 to send all the MSU processing to the same task.

### Default

```
AUTSYS AUTOBASE AUTO1
```

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

### **%AOFLOPMSU%**

#### Usage

This cascade is used to route commands concerning MSUs. If you are not using the standard names for SA OS/390 autotasks, you must change this synonym. It should equate to the same autotask names as %AOFOPMSU%, with the names enclosed in quotation marks. It is used when a literal string is required. It is mainly used on the Satellite SA OS/390 to send all the MSU processing to the same task.

### Default

```
'AUTSYS AUTOBASE AUTO1'
```

### **%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.

### **%AOF SIRTASK%**

#### 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
```

## A Guide to SA OS/390 Automation Table Samples

This assumes that you have not implemented EMCS support. If you do implement EMCS support, you must change this synonym.

### **%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

### **%AFOPTOPOMGR%**

#### Usage

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

#### Default

TPO&DOMAIN.

### **%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' '

---

## Automation Table Fragments

This section documents all the automation table fragments supplied by SA OS/390. These fragments are included in the main NetView message automation table AOFMSG01.

### **AOFMAPPC**

This entry contains automation table statements for the automation of APPC/MVS and ASCH.

**Required**

Only if running APPC/MVS and ASCH. (NOTE: I/O monitoring functions require the use of APPC/MVS and ASCH.)

**Processed Prefixes**

ATB\*, ASB\*

**Suppressed Messages**

ATB002I, ATB007I, ASB052I, ASB053I

## A Guide to SA OS/390 Automation Table Samples

### AOFMGMFH

This entry contains automation table statements for the automation of GMFHS.

#### Required

Only if running GMFHS.

#### Processed Prefixes

DUI\*

#### Suppressed Prefixes

DUI\*

### AOFMIO

This entry contains automation table statements for the automated startup and monitoring of SA OS/390 I/O operations components.

#### Required

If you want to automate the startup of I/O operations.

#### Processed Prefixes

IHV\*, DSI\*

**Suppressed messages:** IHVD000I, IHVD001I, IHVD002I, IHVD003I, AOF330I

### AOFMJES2

This entry contains automation table statements for the automation of JES2.

#### Required

Only if running JES2.

If you are using a prefix character of other than '\$' for your JES2, you must change the prefix character coded in the fragment.

If you are running PolyJes you need one copy of the statements, modified with suitable prefix characters, for each JES2 you are running.

#### Processed Prefixes

\$HASP\*

#### Suppressed Prefixes

\$HASP\*

### AOFMJES3

This entry contains automation table statements for the automation of JES3.

#### Required

Only if running JES3.

### Processed Prefixes

IAT\*

### Suppressed Prefixes

IAT\*

## AOFMMVS

This entry contains automation table statements for the automation of OS/390.

### Required

Yes.

### Processed Prefixes

IEA\*, IEC\*, IEE\*, IEF\*, IFB\*, IFC\*, IXC\*, IXL\*

### Suppressed Prefixes

IEA\*, IEC\*, IEE\*, IEF\*, IFB\*, IFC\*, IXC\*, IXL\*

## AOFMNETV

This entry contains automation table statements for the automation of NetViews.

### Required

Yes.

It provides automation for SA OS/390's host NetView and some traps to improve performance.

### Processed Prefixes

AAU\*, BNH\*, BNJ\*, CNM\*, DFI\*, DSI\*, DWO\*

### Suppressed Prefixes

AAU\*, BNH\*, BNJ\*, CNM\*, DFI\*, DSI\*, DWO\*

## AOFMPOPS

This entry contains automation table statements for the automated startup and monitoring of the SA OS/390 processor operations component.

### Required

If you want to automate the startup of processor operations and monitor the component on the SA OS/390 graphic interface.

### Processed Prefixes

## A Guide to SA OS/390 Automation Table Samples

ISQ\*

**Suppressed messages:**

ISQ091I, ISQ092I

### **AOFMRMF**

This entry contains automation table statements for the automation of RMF and RMF Monitor III.

**Required**

Only if running RMF and, optionally, RMF Monitor III.

**Processed Prefixes**

ERB\*

**Suppressed Prefixes**

ERB\*

### **AOFMRODM**

This entry contains automation table statements for the automation of RODM.

**Required**

Only if running RODM.

**Processed Prefixes**

EKG\*

**Suppressed Prefixes**

EKG\*

### **AOFMSGSY**

This entry defines automation synonyms for usage in the remainder of the automation table.

**Required**

YES

### **AOFMSGGW**

This entry contains automation tables statements dealing with the setup and running of SA OS/390's inter-NetView gateway tasks.

**Required**

If you are using SA OS/390 gateway function.

**Processed Prefixes**

DSI\*, ROUTE, GWREQ

### **AOFMSGST**

This entry contains automation table statements for processing MSUs on the NMC focal point system.

#### **Required**

It should be included only on the NetView that is driving the NMC focal point system.

If activating AOFMSGST on a system that does not contain an alert focal point NetView, you must comment out the SRF filters. The SRF filters prevent heartbeat alerts from flooding NPDA screens on the alert focal point NetView. For an SA OS/390 monitored Alert Intermediate Focal Point the SRF BLOCK statements must be commented out. For all systems you should consider the consequences that these extra alerts will have on the NPDA Hardware monitoring screen before commenting out these filters.

#### **Processed Prefixes**

FLC, MSU

### **AOFMSGTS**

This entry contains automation table statements for processing MSUs originating from processor operations.

#### **Required**

It should be included only on the NetView that is driving the processor operations focal point system.

### **AOFMSGPM**

This entry contains automation table statements to trap messages to be handled by processor operations.

#### **Required**

If you are using SA OS/390 processor operations

#### **Processed Prefixes**

ISQ\*

### **AOFMSGPA**

This entry contains automation table statements for processor operations handling of alerts and messages from OCF-based processors.

#### **Required**

If you are using SA OS/390 processor operations

#### **Processed Prefixes**

## A Guide to SA OS/390 Automation Table Samples

CSAA\*

### **AOFMSGMS**

This entry contains automation table statements for miscellaneous SA OS/390 message processing.

#### **Required**

It must be included on any NetView running SA OS/390 automation.

#### **Processed Prefixes**

AOF\*, DSI\*

### **AOFMSGHI**

This entry contains automation table statements to trap AOF messages and present them, appropriately highlighted, to the user.

#### **Required**

It should be included on any NetView running SA OS/390 automation. If it is omitted performance is impacted (SA OS/390 produces a LOT of AOF messages) and the message hold functions do not work.

#### **Processed Prefixes**

AOF\*

#### **Suppressed Prefixes**

AOF\*

### **AOFMSGBL**

This entry contains automation table statements to trap remote messages. The aim is to exclude all the messages that do not originate on this system, giving an improvement in performance.

#### **Required**

Not required. If used you have to carefully position any local automation that uses remote messages before it. If you are in a PolyJes environment, you may need to modify some of the traps in it to take account of multiple local system names.

#### **Processed Prefixes**

Many

#### **Suppressed Messages**

All remote messages

### **AOFMSGGN**

This entry contains automation table statements to trigger 'generic' status processing from MVS messages. The entries coded trigger the SA OS/390 generic

## A Guide to SA OS/390 Automation Table Samples

routines ACTIVMSG and TERMMSG in response to a variety of MVS messages. Using this fragment removes the need to code individual ACTIVE and FINAL TERMINATION messages for many of your subsystems.

### Required

Not required. If used you may want to modify it to screen out some of your batch jobs with automation table entries. The AOCFILT routine (which is used in AOFMSGGN) is quite efficient, but adding automation table entries to filter out many of your batch jobs would improve performance.

### Processed Prefixes

IEF4\*, IEFC4\*

## AOFMSGNM

This is for the assortment of messages that need to be alerted to the system operations focal point if the NetView Management Console is running.

### Required

Only if running NetView Management Console

### Processed Prefixes

IEA, IEE, ISG, AHL, ILR, IRA

## AOFMSGXC

This entry contains automation table statements for processor operations XCF message automation.

### Required

If you are using SA OS/390 processor operations

### Processed Prefixes

ISQ\*, IXC\*, IEE\*

## AOFMTSO

This entry contains automation table statements for the automation of TSO.

### Required

Only if running TSO.

### Processed Prefixes

IKT\*

### Suppressed Prefixes

IKT\*

## A Guide to SA OS/390 Automation Table Samples

### **AOFMVLLA**

This entry contains automation table statements for the automation of VLF and LLA.

#### **Required**

Only if running VLF or LLA.

#### **Processed Prefixes**

COF\*, CSV\*

#### **Suppressed Prefixes**

COF\*, CSV\*

### **AOFMVTAM**

This entry contains automation table statements for the automation of VTAM.

#### **Required**

Only if running VTAM.

#### **Processed Prefixes**

IST\*

#### **Suppressed Prefixes**

IST\*

### **INGMTEC**

Used for routing of SA OS/390 messages to the Tivoli Enterprise Console.

#### **Required**

Only if message routing to the Tivoli Enterprise Console (TEC) is desired.

#### **Processed Prefixes**

AOF\*

---

## **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, each of which has its own advantages and disadvantages.

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

The following topics and approaches are discussed:

- "Multiple Master Automation Tables" on page 23

- “Black Box”
- “Insert into AOFMSG01” on page 24
- “Insert into Fragments” on page 24
- “Include SA OS/390 Fragments” on page 24

### Multiple Master Automation Tables

Besides AOFMSG01, 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.

#### Advantages

- You are not forced to modify the shipped AOFMSG01 automation table or any of the fragments, except AOFMSGSY.
- It is easy to apply maintenance to SA OS/390 automation table fragments (although you have to watch for new messages).
- It is easy to maintain your entries, because they are independent from SA OS/390 entries.

#### Disadvantages

- None

**Note:**

This approach is highly recommended.

### Black Box

In this instance your primary automation table consists of two statements - a %INCLUDE for a member containing your statements and a %INCLUDE for AOFMSG01. If you do this you must ensure that your statements do not exclude any messages that SA OS/390 entries need.

#### Advantages

- It is easy to apply maintenance to SA OS/390 automation table fragments (although you have to watch for new messages).
- It is easy to maintain your entries, as they are all in the same place.

#### Disadvantages

- If you get the analysis of the messages SA OS/390 needs wrong, you may disable part or all of SA OS/390.
- Performance may be impacted.

#### Variants

Possible variants on this involve splitting your entries into two tables - one for entries that use messages that SA OS/390 excludes and one for entries that do not. You place the first set in front of the AOFMSG01 include, with CONTINUE(Y) coded as appropriate, and place the second set after the include. Performance is a little better, but it is more work to set up and maintain.

## A Guide to SA OS/390 Automation Table Samples

### Insert into AOFMSG01

Divide your local automation statements into separate members, each containing a set of messages related by message ID or function. Place BEGIN/END pairs and CONTINUE(Y)s where appropriate. Add %INCLUDE statements for the separate members to AOFMSG01 at appropriate points.

#### Advantages

- Performance is better and it is easier to shift things around to tune your table.
- You can accept automatic maintenance of SA OS/390 fragments, but you still need to check for message conflicts.

#### Disadvantages

- You have to restructure and divide your automation table.
- Applying maintenance to AOFMSG01 is difficult.

### Insert into Fragments

Divide your automation table by message ID (with maybe a couple of fragments for miscellaneous messages). Edit the SA OS/390 fragments and put %INCLUDES for your statements inside the BEGIN/END pairs in the fragments. Put %INCLUDEs for your miscellaneous entries into AOFMSG01.

#### Advantages

- This gives you the best performance/tuning option.

#### Disadvantages

- You can no longer accept automatic maintenance for SA OS/390 fragments (and you still need to analyze the changes for message conflicts).
- You have to divide and rearrange your automation table.

### Include SA OS/390 Fragments

Given that you have a working table and that SA OS/390 provides encapsulated fragments that contain SA OS/390 functions, you could just add %INCLUDEs for the SA OS/390 fragments into your automation table at appropriate points.

#### Advantages

- Although you have to rework your automation table, the changes are not as severe.
- You can still do performance tuning.
- You still get automatic maintenance of the SA OS/390 fragments (although you must still check for message conflicts).

#### Disadvantages

- No automatic maintenance for AOFMSG01.
- You must be careful to preserve critical sequencing and resolve any message conflicts.

**Note:** If you are going to insert statements into SA OS/390-supplied fragments, you should always use %INCLUDE. This is much easier than placing the entries directly into the SA OS/390-supplied fragments. Editing the SA OS/390 fragments to stop them excluding messages degrades performance.

---

## Chapter 3. 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 multiple ways 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 51 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.

## How to Automate Your Resources

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 89 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

## When SA OS/390 Checks Automation Flags

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 sample AOFMSGGN in ING.SINGNPRM 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. In these situations you can insert additional entries in front of them 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.

## When SA OS/390 Checks Automation Flags

### 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 40.

### 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.

## When SA OS/390 Checks Automation Flags

2. If job TASK23 came down with an S222 error (an operator cancel), it would be treated as a normal termination message and a TERMMMSG 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 TERMMMSG 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. TERMMMSG 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. TERMMMSG 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 MVS, JES, RMF™, TSO, and RODM. Refer to the sample automation tables for more information.



---

## Chapter 4. How to Add a New Application to Automation

This chapter enumerates the steps that are required to have a new application automated and monitored 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, TERMMMSG, 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 1). 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 1. Example of a WTORS Entry

## How to Add a New Application to Automation

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 37, 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 31.

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. This can only be accomplished by using a job name specific entry in the NetView message automation table, therefore any change in the job name requires a corresponding change to that entry.

**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%));
```

## How to Add a New Application to Automation

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.

## How to Add a New Application to Automation

```
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:

## How to Add a New Application to Automation

```
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,

## How to Add a New Application to Automation

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
```

## How to Add a New Application to Automation

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 5. 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 4. How to Add a New Application to Automation" on page 31.

---

### 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 23).

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 AOFMSG01 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:

## How to Add a Message to Automation

- 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 41.

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 7. How to Create Automation Procedures” on page 51. 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*.

## How to Add a Message to Automation

**Note:** For elementary code matching, ISSUECMD and ISSUEREP may be called with CODE= parameters specified. They will then call 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 43

### Messages Issued by an Application or MVS Component

If a message response is a command, add a call to the ISSUECMD generic routine.

**Note:** The NetView message automation table examples shown in this book all have a DOMAINID check coded. If you are putting the entries into your NetView message automation table after the AOFMSGBL sample supplied with SA OS/390 you do not need the DOMAINID check as the sample contains statements which stop messages from other domains progressing through the NetView message automation table.

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:

## How to Add a Message to Automation

```
IF MSGID = 'AHL125A' & DOMAINID = %AOFDOM%
THEN EXEC(CMD('ISSUEREP AUTOTYP=START') ROUTE(ONE %AOFOPWORS%));
```

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 T0 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 Processor Commands” on page 56. 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 processor operations PC consists of the following:

- ISQ900I or ISQ901I message identifier
- Name of the PC where the message originated
- Port letter of the console to which the message was sent
- Message identifier and text of the original message from the target system

For example, if a PC forwards the IEA101A message to the processor operations focal point system, SA OS/390 creates one or both of the following SA OS/390 messages:

```
ISQ900I PS2name port IEA101A SPECIFY SYSTEM PARAMETERS
ISQ901I PS2name port IEA101A SPECIFY SYSTEM PARAMETERS
```

A message forwarded from a NetView 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 console. That message format depends on the console

## How to Add a Message to Automation

definitions. For example, if an OS/390 operator console is defined with MFORM=(T,J), each message contains the time and job number before the message text.

**Note:**

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

If you use backup operator and system consoles, the messages from these backup consoles are not forwarded to the focal point system.

## 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 ISQ055I

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 ISQ055I Unable to unlock *PS/2-name port-id*.

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 ISQ055I 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 ISQ055I 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 PC or a 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 PC or support element. An unsuccessful operation at the target system generates an unsolicited message that the PC or 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.

## How to Add a Message to Automation

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: <b>IEA120A</b> Device ddd volid read, reply cont or wait. <b>IOS120A</b> Device ddd shared (PR volid not read.) the recovery task, reply cont or wait.
INGEI357	ISQI357	Issues the following response to the target: CONT Responds to the following message: <b>IEE357A</b> Reply with SMF values or U.
INGEI426	ISQI426	Issues the following response to the target: U Responds to the following message: <b>\$HASP426</b> Specify options - subsystem_id.
INGEI502	ISQI502	Issues the following response to the target: WARM,NOREQ. Responds to the following message: <b>ICH502A</b> Specify name for primary/backup RACF data set sequence nnn or none.
INGEI877	ISQI877	Issues the following response to the target: NONE Responds to the following message: <b>IEA877A</b> Specify full DASD SYS1.DUMP data sets to be emptied, tape units to be used as SYS1.DUMP data sets or GO.
INGEI956	ISQI956	Issues the following response to the target: GO Responds to the following message: <b>IEE956A</b> 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 AOFCMDPO 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

## How to Add a Message to Automation

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 AOFCDM 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 PC 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.

## How to Add a Message to Automation

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 37.

---

## Chapter 6. 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 the following. A description of their purposes is given in *System Automation for OS/390 Defining Automation Policy*.

- AOFADMN
- AOFAJMON
- AOFATMON
- AOFAPMON
- AOFPCSM
- AOFUXMON
- 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 7. How to Create Automation Procedures

### Programming Interface information

---

This section contains Programming Interface Information.

### End of Programming Interface information

---

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 52 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 2 on page 53 illustrates the structure of automation procedures for system operations and Figure 3 on page 53 illustrates the structure of automation procedures for processor operations.

## How to Create Automation Procedures

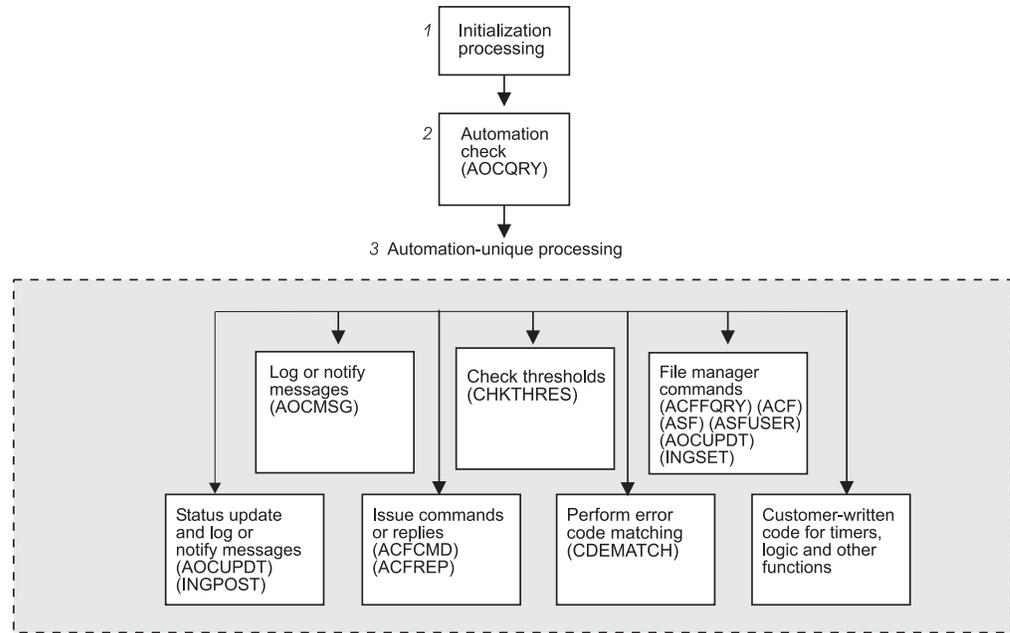


Figure 2. Automation Procedures for System Operations

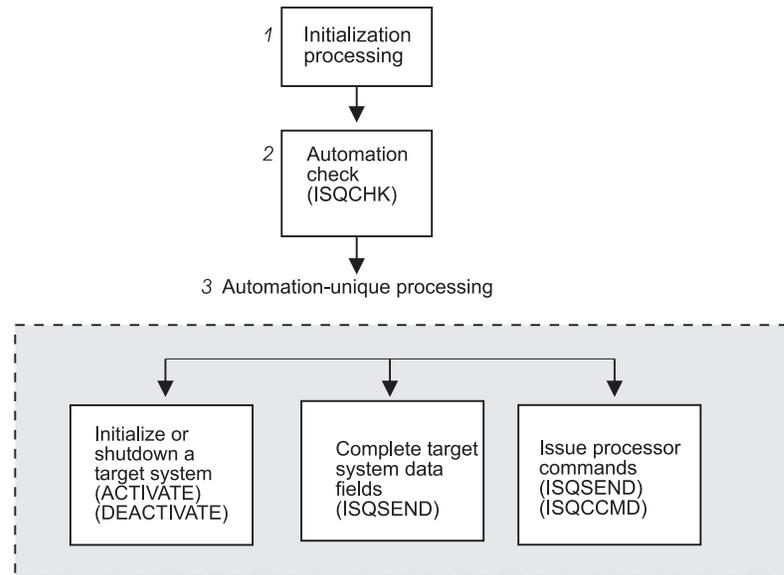


Figure 3. Automation Procedures for Processor Operations

## 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.

## How to Create Automation Procedures

- 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 107 and “Appendix B. Global Variables to Enable Advanced Automation” on page 113 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 62, 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 56

### Automation Processing in System Operations

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

## How to Create Automation Procedures

**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, ISSUECMD 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*.

## How to Create Automation Procedures

**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 collects 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 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 Processor Commands:** SA OS/390 processor operations together with a systems management software such as Netfinity® base services, installed on a remote processor operations PC, provide additional support for system console and operator console operation.

Note, that only system consoles of screen based processors (9021, 9121, 3090, 308x, and 4381) can be connected to a processor operations PC. For the supported system consoles, processor operations does not provide the functions of a service console, only the non-service modes are supported.

From an operator workstation running the Netfinity remote manager services code, you can operate the system and operator consoles manually, if the manager is in session with a processor operations PC as its Netfinity client. Consider this as an alternative to the processor operations pass-through mode, which allows you to operate remote consoles from a central processor operations focal point NetView.

## How to Create Automation Procedures

With processor operations, you can use the OCF commands to communicate with the zSeries or 390-CMOS processors in order to control the hardware. As an alternative for this you may use a HMC (Hardware Management Console). Or you may use the HMC's web-server, which allows you to perform the basic operator tasks with a web-browser from any operator workstation connected to your network.

**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: ISQTCC, 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, ISQMVM11, ISQMVM12, ISQMLCON, ISQMLPS2, ISQMWAIT, ISQMDCCF, ISQMOCOK, ISQMPST, ISQM020, and ISQIPLC.

**Completing Target System Data Fields:** If you are using a PC, you can use your routines to automate any action that a target system operator may be required to perform. This includes placing data in the input fields of a target system console screen. Use the ISQSEND command to send a #DATA command to the console screen handler.

The #DATA command simulates typing the accompanying command string into the specified field. The command includes a parameter to designate the field where the data is to be put.

Assume you want to put the characters A, B, and C into the three input fields of a particular system console panel. Use the following command sequence in your automation routine:

```
ISQSEND Target OC #DATA 1 A
ISQSEND Target OC #DATA 2 B
ISQSEND Target OC #DATA 3 C
ISQSEND Target OC #ENTER
```

The order of the #DATA commands is not significant as long as you issue each of them before the #ENTER command.

**Designating the Current Logical Partition:** If you are operating in a screen-oriented processor environment, target systems running in logical partitions share the same system console port in the PC. If you send information to a system console, be sure that the command is routed to the correct partition.

Use the ISQXSLP command to designate the current logical partition. All commands sent to that system console port are directed to the target system running on the current logical partition.

**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.

## How to Create Automation Procedures

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*.

## How to Create Automation Procedures

**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.

The ISQTCC command has an optional parameter that locks the console. The lock is released automatically when the routine exits the pass-through mode established by the ISQTCC command. This lock can also be overridden by the ISQOVRD command, but you should be sure that you want to override a lock for an operator using the pass-through mode.

---

## 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.

## How to Create Automation Procedures

### 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 4. 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 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*).

## Developing Messages for Your Automation Procedures

- 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 48 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

9  Otherwise Do
     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 67).
- 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.

## Example Automation Procedure

- 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.

## Testing and Debugging Automation Procedures

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
-------	-------------

<b>D (Display)</b>	
--------------------	--

	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:
--	---

- |  |  |
|--|--|
|  | <ul style="list-style-type: none"><li>• Issue the SA OS/390-generated command</li><li>• Change the command</li><li>• End the operation</li></ul> |
|--|--|

**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.

## Testing and Debugging Automation Procedures

**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
- ISSUEREPLY

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.

```

----- DETAIL STATUS DISPLAY -----
                                     1 of 1

COMPONENT      : RMF           SYSTEM   : ATLMVS1
COLOR          : PINK         PRIORITY : 255
DATE           : 06/14/93     TIME     : 12:05:01
REPORTER       : GATCNM01     NODE     : ATL01

REFERENCE VALUE: RMF

AOF317A 12:04 : ASSIST DISPLAY FROM CNM01/AUTWRK01 - FOR:
SUBSYSTEM/RMF - KEY: RMF/SHUTDOWN/PASS1 - CMD: MVS P RMF

===>
1-HELP      3-RETURN 4-DELETE   6-ROLL 7-UP 8-DOWN 9-ASSIST 11-BOTTOM 12-TOP
```

Figure 5. 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

## Testing and Debugging Automation Procedures

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.

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.

## Coding Your Own Information in the Automation Status File

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.

## Coding Your Own Information in the Automation Status File

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
- EVI
- EVE
- EVJ

## Coding Your Own Information in the Automation Status File

---

## Chapter 8. How to Automate Processor Operations Controlled Resources

This chapter contains information on 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 on how to achieve such message automation.

“Message Automation for IXC102A” on page 77 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.

## How to Automate Processor Operations Controlled Resources

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 now 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).

Now 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:

<p><b>MVS SYSNAME = ProcOps name</b> The <i>MVS SYSNAME</i> must be identical with the <i>ProcOps name</i>.</p>
---

If this is not the case, you need to change it subsequently.

- 2.

<p><b>Job Name = ProcOps name</b> The <i>Job Name</i> 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.</p>
--

- 3.

## How to Automate Processor Operations Controlled Resources

### 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
```

## How to Automate Processor Operations Controlled Resources

---

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.

For regular connections processor operations PCs forward messages from target systems to a focal point NetView. Alternatively when using Console Integration the CMOS Processor's Support Element forwards message data blocks (MDBs) from target systems as CSAxxxx messages to a focal point NetView. The CSAA-Messages will be trapped in the MAT and an ISQ900I message will be built by processor operations.

**Note:**

If a target system connection is configured via a processor operations PC as well as via Console Integration, only one connection should be active to avoid messages being sent through both connections. This may result in automating a WTO message twice.

ISQ900I 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.

### MFORM() parameter in CONSOLxx member

The target system message is sent in the same format as it would be displayed on the console. That message format depends on the console definitions. For example, if an MVS operator console is defined with MFORM=(T,J), each message contains the time and job number before the message text. Make sure your consoles are operating as you expect and write your NetView message automation table to respond to the correct message format. For adding an entry in the NetView message automation table please refer to “Chapter 5. How to Add a Message to Automation” on page 39.

### Buffered messages in Processor Operations PC

Once the communication path via a Processor Operations PC has been established, it will forward all target system messages to the SA OS/390 processor operations focal point system. When the communication path is closed (e.g. using an ISQXCLS command) all target system messages are buffered in the Processor Operations PC. When the communication path is re-initialized (using the ISQXIII command) all buffered messages will be forwarded to the focal point system. This may result in automating messages which are already obsolete.

### Sample NetView Message Automation Table AOFMSGXC

This NetView message automation table fragment checks for ISQ900I and certain message id's contained in the ISQ900I. In case of a match ISSUECMD, ACTIVMSG, TERMMMSG 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 AOFMSGPM.

### 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.

## How to Automate Processor Operations Controlled Resources

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 6.

```

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 6. Message Processing Panel

Pressing ENTER will lead you to panel *CMD Processing* Figure 7 on page 79.

3. In our sample from Figure 7 on page 79, 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 79.

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.

## How to Automate Processor Operations Controlled Resources

```

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 7. CMD Processing Panel

Pressing PF3 (END) will lead you back to panel *Message Processing* from Figure 6 on page 78.

- Now you enter **code** into the *Action* column of panel *Message Processing*. This will lead you to the *Code Processing* panel as shown in Figure 8.

```

COMMANDS  HELP
-----
AOFPISSC          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 8. 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,

## How to Automate Processor Operations Controlled Resources

LOADCLEAR, RESETFORCE, LOADFORCE) shown in the *CMD Processing* panel (Figure 7 on page 79). 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 9.

**Note:** The *Major Resource* field from Figure 9 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 9. 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.

## How to Automate Processor Operations Controlled Resources

```
-----  
A0FGFAS1          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 10. 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 9. 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 93.
4. An exit for INGREQ processing: AOFEXC01; this is also a static exit described in "Static Exits" on page 84.

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:

## SA OS/390 User Exits

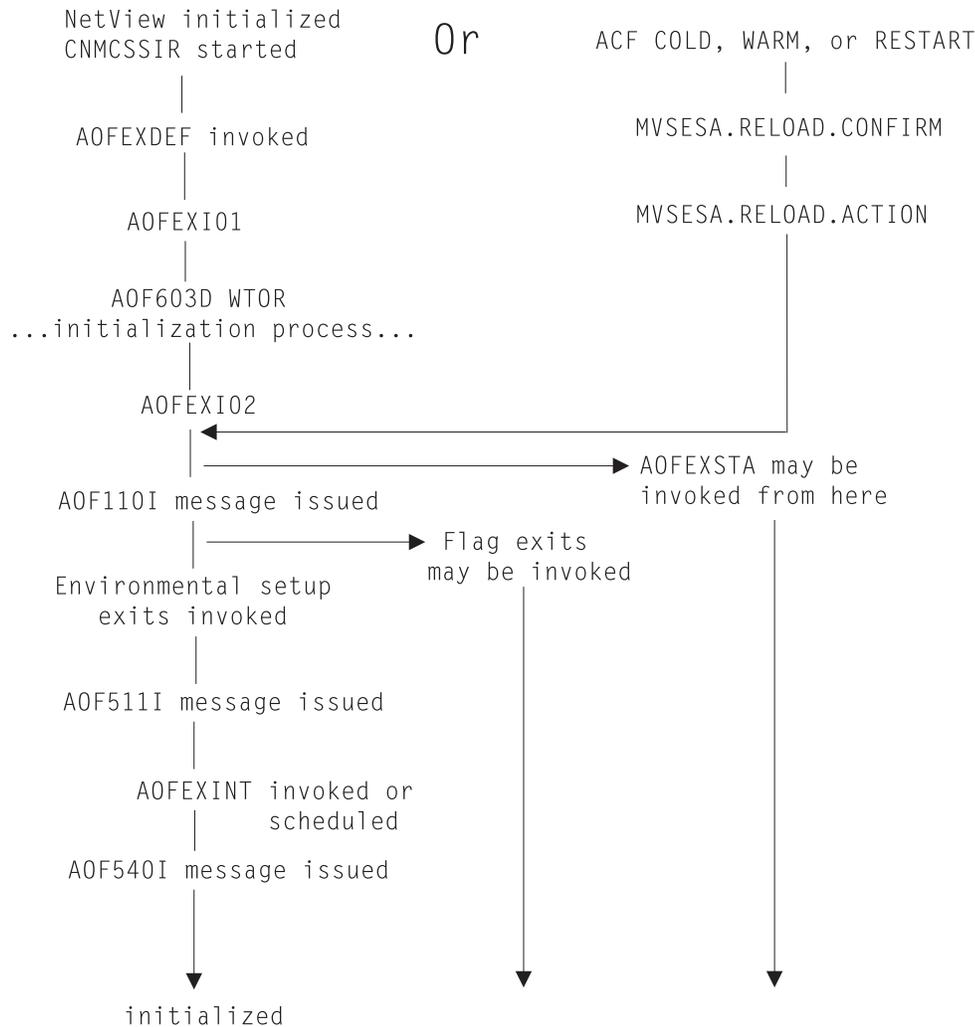


Figure 11. SA OS/390 Exit Sequence during SA OS/390 Initialization

## 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 113 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.

## 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 AOFSEXINT in “Appendix B. Global Variables to Enable Advanced Automation” on page 113.

**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.

## SA OS/390 User Exits

**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, and so on).

**Note:**

Consider using ISSUEREPL, 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

## SA OS/390 User Exits

### APPL\_PARMS

specifies the application parameters

**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

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.

## 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

## SA OS/390 User Exits

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, and so on 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.

## SA OS/390 User Exits

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 TGPFIX 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 TGPFIX value. You cannot use SUB. The TGPFIX value you specify should be different from the TGPFIX parameter you were passed. You are responsible for ensuring the uniqueness of all TGPFIXs 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 94
- “User Exits for DELETE Processing” on page 94
- “User Exits for CONVERT Processing” on page 95

“Invocation of Customization Dialog Exits” on page 95 provides information on what to do to activate the user exits.

## User Exits for BUILD Processing

Two user exits are implemented for the process of building the automation control file (BUILDF).

1. INGEX01, which is called before the automation control file build function starts.
2. INGEX02, which is called after the automation control file build function (BUILDF) has ended.

The following parameters are passed to both INGEX01 and INGEX02 exits, separated by commas:

- Parm1 = PolicyDB name
- Parm2 = Enterprise name
- Parm3 = BUILD output dataset
- Parm4 = entry type (or blank)
- Parm5 = entry name (or blank)
- Parm6 = BUILD type (MOD/ALL)
- Parm7 = BUILD mode (ONLINE/BATCH)
- Parm8 = Sysplex name (or blank)
- Parm9 = Build option (1,2, or 3)

## SA OS/390 User Exits

If user exit INGEX01 produces return code RC = 0, the BUILDf processing continues. If a return code RC > 0 is produced, an error message will be given and the BUILDf processing will terminate.

If user exit INGEX01 ended with return code RC > 0, user exit INGEX02 will not be called (as BUILDf will not start) and the processing terminates.

User exit INGEX02 will always be called once 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 will be displayed.

## 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.

## SA OS/390 User Exits

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.

---

## Chapter 10. Maintaining the Processor Operations PC

SA OS/390 keeps processor operations PC configuration data in as many as four types of data sets:

- ISPF table library data set used by the SA OS/390 customization dialog  
This data set provides the values displayed in the customization dialog panels each time you start the customization dialog. It is updated each time you make changes during that dialog session.
- SA OS/390 control file on the focal point system  
This is the data set SA OS/390 processor operations uses for configuration data when it is running. It is updated by invoking option 2 on the Build Functions menu.
- processor operations PC profile data set on the focal point system  
This partitioned data set has one profile (member) for each SA OS/390 processor operations PC in your configuration. The entire data set is updated whenever you select option 5 to build processor operations PC profiles (Build Functions Menu). You can also update the data set using an editor, such as the editor in ISPF.
- PS/2 file on each PS/2 computer to be run using the stand-alone function  
This file is a copy of the corresponding member in the PS/2 profile data set on the focal point system. It is updated by downloading the member from the focal point system. You can also update the file using an editor, such as the OS/2<sup>®</sup> system editor.

In general, all of these data sets should be kept in synchronization with each other. However, you may have situations when it is appropriate for one or more of the data sets to be mismatched from the others for a time, such as when you are introducing changes into your enterprise configuration. Once the configuration data is out of synchronization, you should know how to restore synchronization efficiently. The following sections address this topic.

---

### Editing the Profile Data Set

You can use an editor to create new PS/2 profiles or to modify existing ones. These may be either the members of the profile data set on the focal point system or the PS/2 profile files on the processor operations PS/2 computers. For reasons discussed below, it is usually better to edit the profiles on the PS/2 computer rather than the profiles on the focal point system.

If you have installed the standalone function, a sample PS/2 profile is available in the ING.INGSAMP library of SA OS/390. Member INGSAMPL contains the port definition statements, the configuration commands for all supported console types, and the external clock startup command.

**Note:** INGSAMPL is not a ready-to-use PS/2 profile. Make a copy and customize the copy to meet your SA OS/390 installation requirements.

If you edit a member of the profile data set, make sure that the records have no line numbers. If you use the ISPF editor, set the ISPF edit profile of that member to NUMBER OFF.

## Maintaining the Processor Operations PC

The PS/2 profile may contain port definition statements for active and backup console connections. For backup console connections, the screen handler function that routes console messages to the focal point system should be disabled. Each PS/2 profile member has an entry similar to the following:

```
*PORT QUEUE E DISABLE MESSAGE
```

This is a screen handler function DISABLE command statement, marked as a comment. If port E in this example is defined as a backup console connection, you should remove the asterisk (\*) in column 1 of the statement line to obtain:

```
PORT QUEUE E DISABLE MESSAGE
```

The statement can now be executed. It disables all messages from this console port being routed to the focal point system.

If you want to customize the profile on the processor operations PS/2 computer after a PS/2 profile has been downloaded, you can use an OS/2 editor to make the changes.

**Attention:** Do not modify the ISQXPC.PRO file in subdirectory ISQPS2.

---

## Keeping PS/2 Configuration Data Synchronized

It is important to match the processor operations PS/2 configuration data in all the data sets where it is maintained. Mismatches can cause a variety of failures or spurious results, particularly during a switch-over from operation with the standalone function to normal operation with a focal point system.

Keeping port letters synchronized between the PS/2 profiles and the focal point system control file is essential. If the port letters do not match, you may have to enter additional commands manually before the focal point system can assume control of the target systems attached to the PS/2 computer. Even worse, a mismatch in the focal point port or PS/2 control port letters can prevent the focal point system from establishing control of the processor operations PS/2 computer without manual intervention at the PS/2 computer.

When you have mismatched PS/2 configuration data and you want to restore synchronization, the procedure to follow depends on which data set has the "master" data. In this discussion, the "master" copy of the data is defined as the *one* copy that you want to propagate to all the other data sets. Because you can modify the PS/2 configuration data either by using the customization dialog or by editing, the master copy of the data can be in the ISPF tables or in the PS/2 profile data sets on either the focal point system or the PS/2 computer. It is unlikely that the control file has the master copy, because it is difficult to update the control file without concurrently updating the ISPF tables.

Table 4 on page 99 outlines how to bring each of the other PS/2 configuration data sets into synchronization with a particular master copy.

Table 4. Synchronizing PS/2 Configuration Data

"Master" data set	To bring other configuration data into synchronization with master...			
	ISPF tables	Control file	Profile on focal point system	Profile on PS/2 computer
ISPF tables	Not applicable	Run Option 4 from Build Functions menu	Display "SA OS/390 Build Functions Menu". Select option 6 to build PS/2 profile information and perform the build	Download from updated profile on focal point system
Profile on focal point system	Reenter data into customization dialog	Reenter data into customization dialog, then Option 4 from Build Functions menu	Not applicable	Download from focal point system
Profile on PS/2 computer	Reenter data into customization dialog	Reenter data into customization dialog, then Option 4 from Build Functions menu	Reenter data into customization dialog. Display "SA OS/390 Build Functions Menu". Select option 6 to build PS/2 profile information and perform the build	Download from updated profile on focal point system
Not applicable				

For information on how the commands in a PS/2 profile data set relate to entries in the configuration dialog panels, refer to "PS/2 Configuration Data Cross-Reference"

The table above indicates why, if you want to edit a PS/2 profile data set, it is preferable to change the PS/2 file rather than the focal point system data set. That way, as you re-synchronize the data sets, you can easily compare the newly updated focal point system data set with the master copy on the PS/2 computer. This helps you validate that you correctly entered the data into the customization dialog.

---

## PS/2 Configuration Data Cross-Reference

Use Table 5 on page 100 to associate commands in a PS/2 profile with PS/2 definition entries in the configuration dialogs. Table 5 on page 100 may be most useful if you have created or modified a PS/2 profile using an editor and now need to use the SA OS/390 customization dialogs to synchronize the profile and the processor operations control file on the focal point system.

Sometimes a profile command is the result of a single entry (a value supplied for a field on a configuration dialog panel), but more commonly a profile command comes from two or more dialog entries. This table focuses on the configuration dialog entries that set variable parameters in the PS/2 profile commands.

When you use the configuration dialogs to change PS/2 definitions, you usually need to make the corresponding changes in the target hardware and target system definitions. This is necessary to maintain consistency throughout the SA OS/390 customization data.

## Maintaining the Processor Operations PC

Table 5. PS/2 Data Cross-Reference

PS/2 Profile Contents		Configuration Dialog Entries		
Command and Usage	Panel Name	Panel ID	Field(s)	
<p>DEFINE PC x</p> <p>This command defines the PS/2 control port. It must be the first command in the profile.</p>	SA OS/390 Processor Operations PS/2 Characteristics	ISQDPP05	PS/2 control port letter => x	
<p>DEFINE TARGET x CUTn</p> <p>This command associates a target port with a 3270 emulator adapter card.</p>	SA OS/390 3270 Port Characteristics	ISQDPP06	Port letter => x	
<p>DEFINE TARGET x LINEn</p> <p>This command makes an association between a 3215 TPF operator console port and an ASCII adapter card.</p>	SA OS/390 Processor Operations Port Configurations	ISQDPP01	<p>Action Add =&gt; Type Address/Number N</p> <p>N is FIRST, SECOND, THIRD, or FOURTH and corresponds to CUTn, where n is 1, 2, 3, or 4.</p>	
<p>DEFINE TARGET x FS_COMn</p> <p>This command makes an association between a 3151 system console port for an ES/9000® processor and an ASCII adapter card.</p>	SA OS/390 ASCII Port Characteristics	ISQDPP07	Port letter => x	
	SA OS/390 Processor Operations Port Configurations	ISQDPP01	<p>Action Add =&gt; Type Address/Number COMn</p> <p>The value n in COMn corresponds to LINEn, where n is 1, 2, or 3.</p>	
	SA OS/390 ASCII Port Characteristics	ISQDPP07	Port letter => x	
	SA OS/390 Processor Operations Port Configurations	ISQDPP01	<p>Action Add =&gt; Type Address/Number COMn</p> <p>The value n in COMn corresponds to FS_COMn, where n is 1, 2, or 3.</p>	

Table 5. PS/2 Data Cross-Reference (continued)

PS/2 Profile Contents		Configuration Dialog Entries		
Command and Usage	Panel Name	Panel ID	Field(s)	
<p>PORT QUEUE x CLOCK STOP PORT QUEUE x CLOCK START COMn bbbbb d s p</p> <p>These commands start the external (WWV) clock routine.</p>	<p>SA OS/390 Processor Operations PS/2 Characteristics</p> <p>SA OS/390 WWV Information</p>	<p>ISQDPPP05</p> <p>ISQDPPP08</p>	<p>PS/2 control port letter =&gt; x WWV installed =&gt; YES</p> <p>COM address =&gt; n Baud rate =&gt; bbbbb Data length =&gt; d Parity =&gt; PPPPP Number of stop bits =&gt; s</p> <p>Note that the value <i>PPPPP</i> on panel ISQDPPP08 maps to the parameter <i>p</i> in the profile command according to the following:</p> <p>P PPPPP - ---- 0 NONE 1 ODD 2 EVEN 3 MARK (parity bit is always 1) 4 SPACE (parity bit is always 0)</p>	
<p>PORT QUEUE x DISABLE MESSAGE</p> <p>This command disables a backup console from delivering messages to the focal point system.</p>	SA OS/390 3270 Port Characteristics	ISQDPPP06	Port letter => x	
<p>PORT QUEUE x ENABLE</p> <p>This command enables all screen handler functions on a console port.</p>	SA OS/390 ASCII Port Characteristics	ISQDPPP07	Port letter => x	
<p>PORT QUEUE x PORT START</p> <p>This command is used to start system console ports for all target systems.</p>	SA OS/390 3270 Port Characteristics	ISQDPPP06	Port letter => x	
<p>PORT QUEUE x PORT START CLEAR</p> <p>This command is used to clear the screen of all 3270 operator console ports for target systems.</p>	SA OS/390 ASCII Port Characteristics	ISQDPPP07	Port letter => x	
	SA OS/390 3270 Port Characteristics	ISQDPPP06	Port letter => x	

## Maintaining the Processor Operations PC

Table 5. PS/2 Data Cross-Reference (continued)

PS/2 Profile Contents		Configuration Dialog Entries		
Command and Usage	Panel Name	Panel ID	Field(s)	
<p>PORT QUEUE x PORT START POWERON</p> <p>This command is used to issue a power-on-reset to an ASCII (TPE) target port.</p>	SA OS/390 ASCII Port Characteristics	ISQDPPP07	Port letter => x	
<p>PORT QUEUE x SET ATTN hh</p> <p>This command sets the 2-hexadecimal-digit ATTN code for an ASCII or serial port.</p>	SA OS/390 ASCII Port Characteristics	ISQDPPP07	Port letter => x ATTN code => hh	
<p>PORT QUEUE x SET ENDOFLINE hh hh hh hh</p> <p>This command sets the 2-to-8-hexadecimal-digit end-of-message code for an ASCII or serial port.</p>	SA OS/390 ASCII Port Characteristics	ISQDPPP07	Port letter => x End of message code => hhhhhhh	
<p>PORT QUEUE x SET ENTER hh</p> <p>This command sets the 2-hexadecimal-digit Enter code for an ASCII or serial port.</p>	SA OS/390 ASCII Port Characteristics	ISQDPPP07	Port letter => x Enter code => hh	
<p>PORT QUEUE x SET LINE bbbbbb d s p</p> <p>This command sets the asynchronous line characteristics for an ASCII or serial port.</p>	SA OS/390 ASCII Port Characteristics	ISQDPPP07	Port letter => x Baud rate => bbbbbb Data length => d Parity => ppppp Number of stop bits => s	<p>Note that the value <i>ppppp</i> on panel ISQDPPP07 maps to the parameter <i>p</i> in the profile command according to the following:</p> <p>p    ppppp -    ----- 0    NONE 1    ODD 2    EVEN 3    MARK (parity bit is always 1) 4    SPACE (parity bit is always 0)</p>
<p>PORT QUEUE x SET MANUAL ON</p> <p>This command sets the force manual switch for a 4381 system console.</p>	SA OS/390 3270 Port Characteristics	ISQDPPP06	Port letter => x	

Table 5. PS/2 Data Cross-Reference (continued)

PS/2 Profile Contents		Configuration Dialog Entries		
Command and Usage	Panel Name	Panel ID	Field(s)	
<p>PORT QUEUE x SET RETRY n</p> <p>This command sets the number of times that the application will attempt to send something to a target system.</p>	SA OS/390 3270 Port Characteristics	ISQDPPP06	Port letter Inbound retry count => n	
<p>PORT QUEUE x SET SCREEN SWITCH FORCE zzzz</p> <p>This command forces a 3270 system or operator console port to switch to the indicated screen handler.</p>	SA OS/390 ASCII Port Characteristics	ISQDPPP07	Port letter Inbound retry count => n	
<p>PORT QUEUE x SET SCREEN SWITCH FORCE</p> <p>This command forces a 3270 system or operator console port to switch to the indicated screen handler.</p>	SA OS/390 3270 Port Characteristics	ISQDPPP06	Port letter Console type => x => zzzz	
<p>PORT QUEUE x SET SCREEN SWITCH FORCE SYS9121</p> <p>This command forces an ASCII system console port on an ES/9000 processor to switch to the indicated screen handler.</p>	SA OS/390/ASCII Port Characteristics	ISQDPPP07	Port letter => x	
<p>PORT QUEUE x SET SCREEN TYPE zzzz</p> <p>This command identifies a type of 3270 system or operator console screen handler to be supported at the indicated port.</p>	SA OS/390 3270 Port Characteristics	ISQDPPP06	Port letter => x Console type => zzzz	
<p>PORT QUEUE x SET SCREEN TYPE SYS9121</p> <p>This command identifies the type of ASCII system console screen handler to be supported at the indicated port for an ES/9000 processor.</p>	SA OS/390 ASCII Port Characteristics	ISQDPPP07	Port letter => x	
<p>PORT QUEUE x SET TERMINAL TYPE zzzz</p> <p>This command identifies the model type for a 3270 system or operator console.</p>	SA OS/390 3270 Port Characteristics	ISQDPPP06	Port letter => x Console model => zzzz	
<p>PORT QUEUE x SET TERMINAL TYPE MOD2</p> <p>This command identifies the model type for an ASCII system console.</p>	SA OS/390 ASCII Port Characteristics	ISQDPPP07	Port letter => x	

## Maintaining the Processor Operations PC

Table 5. PS/2 Data Cross-Reference (continued)

PS/2 Profile Contents		Configuration Dialog Entries		
Command and Usage	Panel Name	Panel ID	Field(s)	
<p>PORT QUEUE x SET TIMEOUT hh</p> <p>This command sets the 2-hexadecimal-digit timeout code for an ASCII or serial port.</p>	SA OS/390 ASCII Port Characteristics	ISQDPPP07	Port letter => x Timeout code => hh	
<p>PORT QUEUE x SET TIMER DELETE nn</p> <p>This command sets the maximum number of seconds to wait after a screen change before again clearing an MVS or VSE operator console screen.</p>	SA OS/390 3270 Port Characteristics	ISQDPPP06	Port letter => x MVS/VSE console interval => nn	
<p>PORT QUEUE x SET TIMER POLL nnnn</p> <p>This command sets the interval the PS/2 computer is to wait before polling to verify that a target port is still responding.</p>	SA OS/390 3270 Port Characteristics	ISQDPPP06	Port letter => x Screen handler poll timer => nnnn	
<p>PORT QUEUE x SET TIMER READ nnnn</p> <p>This command sets the interval that a VM READ or CP READ can stay on a VM console before a message is generated instructing you to take appropriate action.</p>	SA OS/390 ASCII Port Characteristics	ISQDPPP07	Port letter => x Screen handler poll timer => nnnn	
<p>PORT QUEUE x SET TIMER RESPONSE n</p> <p>This command sets the interval the PS/2 computer is to wait for a target console to accept input.</p>	SA OS/390 3270 Port Characteristics	ISQDPPP06	Port letter => x VM console interval => nnnn	
<p>PORT QUEUE x SET TRANSLATE 37 nn</p> <p>This command sets the translation tables to be used in the PS/2 computer for a 3270 console port.</p>	SA OS/390 3270 Port Characteristics	ISQDPPP06	Port letter => x Response timer => n	
<p>Port letter Control unit country code =&gt; nn</p>		ISQDPPP07	Port letter => x Response timer => n	
<p>PORT QUEUE x SET TRANSLATE 437 nn</p> <p>This command sets the translation tables to be used in the PS/2 computer for an ASCII (TPF) console port.</p>	SA OS/390 ASCII Port Characteristics	ISQDPPP07	Port letter => x Control unit country code => nn	

Table 5. PS/2 Data Cross-Reference (continued)

PS/2 Profile Contents		Configuration Dialog Entries		
Command and Usage	Panel Name	Panel ID	Field(s)	
<p>PORT QUEUE x SET XOF hh</p> <p>This command sets the 2-hexadecimal-digit code that indicates the PS/2 computer buffer receiving communications from the target system is full. It applies to ASCII and serial ports.</p>	SA OS/390 ASCII Port Characteristics	ISQDPP07	Port letter => x XOFF code => hh	
<p>PORT QUEUE x SET XON hh</p> <p>This command sets the 2-hexadecimal-digit code that indicates the PS/2 computer buffer is ready to receive communications from the focal point system. It applies to ASCII and serial ports.</p>	SA OS/390 ASCII Port Characteristics	ISQDPP07	Port letter => x XON code => hh	
<p>SET ACTIVE FOR x z</p> <p>This command identifies the PS/2 control port as the active port for the indicated system or operator console.</p>	SA OS/390 3270 Port Characteristics	ISQDPP06	Port letter => x	
	SA OS/390 ASCII Port Characteristics	ISQDPP07	Port letter => x	
	SA OS/390 Processor Operations PS/2 Characteristics	ISQDPP05	PS/2 control port letter => z	

## Maintaining the Processor Operations PC

## Appendix A. Table of External Global Variables

### Programming Interface information

This section contains Programming Interface Information.

### End of Programming Interface information

**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 6. Externalized Common Global Variables

Variable Name	Description	Class	Reference
AOF.clst.ODEBUG	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.clst.OTRACE	Contains a REXX trace setting to be used by the CLIST <i>clist</i> .	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 113
AOFAOCCLONEn	Where <i>n</i> 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 <i>System</i> policy object in <i>System Automation for OS/390 Defining Automation Policy</i>
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 113
AOFCNMASK	The characters which are used in determining unique console names.	3	see "Appendix B. Global Variables to Enable Advanced Automation" on page 113
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 113
AOFDEBUG	Contains a REXX trace setting to be used globally.	2	see <i>System Automation for OS/390 Planning and Installation</i>
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 113
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 113

## Table of External Global Variables

Table 6. Externalized Common Global Variables (continued)

Variable Name	Description	Class	Reference
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 113
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 <i>term/help panel</i> 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 113
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 <i>term/help panel</i> 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 113
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 113
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 113
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 113
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 113
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 <i>System</i> policy object in <i>System Automation for OS/390 Defining Automation Policy</i>

## Table of External Global Variables

Table 6. Externalized Common Global Variables (continued)

Variable Name	Description	Class	Reference
AOF_PRODLVL	Contains the release level of AOC/MVS or SA OS/390. <ul style="list-style-type: none"> <li>For SA/MVS Release 2 the value is SA/MVS, V1R2M0.</li> <li>For SA OS/390 1.3, the value is SA OS/390, V1R3M0.</li> <li>For SA OS/390 2.1, the value is SA OS/390, V2R1M0.</li> </ul>	1	
AOFINITREPLY	Allows you to skip the initial reply AOF603D.	3	see "Appendix B. Global Variables to Enable Advanced Automation" on page 113
AOFJESPREFIX	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 113
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 <i>System Automation for OS/390 Defining Automation Policy</i> , also see "Appendix B. Global Variables to Enable Advanced Automation" on page 113
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 113
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 113
AOFQUICKWTOR	Do not trust the CGLOBALS for WTOR reply processing.	3	see "Appendix B. Global Variables to Enable Advanced Automation" on page 113
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 113
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 113
AOFRESTARTDELAY	The number of seconds that SA OS/390 will use to delay the scheduling of a restart of a subsystem.	3	see "Appendix B. Global Variables to Enable Advanced Automation" on page 113
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 113

## Table of External Global Variables

Table 6. 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 113
AOFSEXINT	Determines how the exit AOFEXINT is processed.	3	see "Appendix B. Global Variables to Enable Advanced Automation" on page 113
AOFSETSTATEOVERRIDE	Sets the default OVERRIDE value for the SETSTATE command.	3	see "Appendix B. Global Variables to Enable Advanced Automation" on page 113
AOFSETSTATESCOPE	allows you to override the predefined default for the SCOPE parameter of the SETSTATE command as documented in <i>System Automation for OS/390 Operator's Commands</i> .	3	see "Appendix B. Global Variables to Enable Advanced Automation" on page 113
AOFSETSTATESTART	allows you to override the predefined default for the START parameter of the SETSTATE command as documented in <i>System Automation for OS/390 Operator's Commands</i> .	3	see "Appendix B. Global Variables to Enable Advanced Automation" on page 113
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 113
AOFSHUTCHK	Sets the default PRECHECK parameter for the SHUTSYS command.	3	see "Appendix B. Global Variables to Enable Advanced Automation" on page 113
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 113
AOFSHUTOVERRIDE	Will set the default OVERRIDE value for the INGREQ command.	3	see "Appendix B. Global Variables to Enable Advanced Automation" on page 113
AOFSHUTSCOPE	Sets the default SCOPE parameter for the SHUTSYS command.	3	see "Appendix B. Global Variables to Enable Advanced Automation" on page 113
AOFSPoolFULLCMD	Determines whether SA OS/390 is to issue Spool Full recovery passes more than once.	3	see the description of the <i>Application</i> policy object in <i>System Automation for OS/390 Defining Automation Policy</i>
AOFSPoolSHORTCMD	Determines whether SA OS/390 is to issue Spool Short recovery passes more than once.	3	see the description of the <i>Application</i> policy object in <i>System Automation for OS/390 Defining Automation Policy</i>
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 113
AOFSUBSYS	The subsystem name of the primary scheduling subsystem.	1	
AOFSYSNAME	Contains the name of the system.	1	See AOCUPDT in <i>System Automation for OS/390 Programmer's Reference</i>

## Table of External Global Variables

Table 6. Externalized Common Global Variables (continued)

Variable Name	Description	Class	Reference
AOFSYSTEM	Contains the system type (MVSESA) as defined in the customization dialog.	1	The <i>Environment Setup</i> panel of the customization dialog.
INGAUTO_INTERVAL	sets the default for the INTERVAL parameter of the INGAUTO command.	3	See INGAUTO in <i>System Automation for OS/390 Operator's Commands</i>
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 <i>System Automation for OS/390 Operator's Commands</i>
INGREQ_OVERRIDE	Sets the default OVERRIDE parameter of the INGREQ command to the specified value.	3	See INGREQ in <i>System Automation for OS/390 Operator's Commands</i>
INGREQ_PRECHECK	Sets the default PRECHECK parameter of the INGREQ command to the specified value.	3	See INGREQ in <i>System Automation for OS/390 Operator's Commands</i>
INGREQ_PRI	Sets the default priority (PRI parameter) of the INGREQ command to the specified value.	3	See INGREQ in <i>System Automation for OS/390 Operator's Commands</i>
INGREQ_REMOVE	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.	3	See INGREQ in <i>System Automation for OS/390 Operator's Commands</i>
INGREQ_RESTART	Sets the default for the RESTART parameter of the INGREQ command when shutting down the resource.	3	See INGREQ in <i>System Automation for OS/390 Operator's Commands</i>
INGREQ_SCOPE	Sets the SCOPE parameter of the INGREQ command to the specified value.	3	See INGREQ in <i>System Automation for OS/390 Operator's Commands</i>
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 <i>System Automation for OS/390 Operator's Commands</i>
INGREQ_TYPE	Sets the default startup/shutdown type (TYPE parameter) of the INGREQ command to the specified value.	3	See INGREQ in <i>System Automation for OS/390 Operator's Commands</i>
INGREQ_VERIFY	Sets the default VERIFY parameter of the INGREQ command to the specified value.	3	See INGREQ in <i>System Automation for OS/390 Operator's Commands</i>

## Table of External Global Variables

Table 6. Externalized Common Global Variables (continued)

Variable Name	Description	Class	Reference
INGSET_VERIFY	Sets the default VERIFY parameter of the INGSET command to the specified value.	3	See INGSET in <i>System Automation for OS/390 Operator's Commands</i> and "Appendix B. Global Variables to Enable Advanced Automation" on page 113
INGVOTE_VERIFY	Sets the default VERIFY parameter of the INGVOTE command to the specified value.	3	See INGVOTE in <i>System Automation for OS/390 Operator's Commands</i> and "Appendix B. Global Variables to Enable Advanced Automation" on page 113
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 <i>Timeout Settings</i> policy object in <i>System Automation for OS/390 Defining Automation Policy</i>
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 <i>Timeout Settings</i> policy object in <i>System Automation for OS/390 Defining Automation Policy</i>
<p><b>Note:</b> There are three different classes of variables in this table, based on the level of access available to the programmer:</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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.</li> <li>• Class 3: Read/write variables. These variables may be set by your own code. It is recommended that you use the <i>AOFEXDEF</i> exit to assign a value to these CGLOBALS.</li> </ul>			

## Appendix B. Global Variables to Enable Advanced Automation

### Programming Interface information

This section contains Programming Interface Information.

### End of Programming Interface information

Table 7 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 7. 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.

## Global Variables to Enable Advanced Automation

Table 7. Global Variables to Enable Advanced Automation (CGLOBALS) (continued)

Variable	Value	Effect
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.
AOFCNMASK		<p>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.</p> <pre> left(opid(),8)  right(opid(),8),   left(aofsysname,4)  right(aofsysname,4),   left(applid(),8)  right(applid(),8),  'ABCDEFGHJKLMNOPQRSTUVWXYZ0123456789\$&amp;#155;#@_!?' </pre> <p>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</p> <p>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).</p> <p>For example:</p> <p>AOFCNMASK: 2A01020304051718</p> <p>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.</p>
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 <i>term/help panel</i> 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.

## Global Variables to Enable Advanced Automation

Table 7. Global Variables to Enable Advanced Automation (CGLOBALS) (continued)

Variable	Value	Effect
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'.
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 <i>System Automation for OS/390 Defining Automation Policy</i> .  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.

## Global Variables to Enable Advanced Automation

Table 7. Global Variables to Enable Advanced Automation (CGLOBALS) (continued)

Variable	Value	Effect
AOFPAUSE	0 to 5	<p>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.</p> <p>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.</p> <p>0 is the default setting.</p>
AOFQUICKWTOR	0	<p>Before a WTOR reply is responded to, an MVS display command is issued to check if it is outstanding.</p> <p>0 is the default setting.</p>
	1	Trust the CGLOBALS for WTOR reply processing.
AOFRELOADOPT	No	<p>Will ignore the Start on RECYCLE option during an ACF reload.</p> <p>This is the default.</p>
	Yes	<p>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.</p>
AOFRESTARTALWAYS	1	<p>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.</p> <p>1 is the default setting.</p>
	0	<p>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.</p>
AOFRESTARTDELAY	1 to 59	<p>This is the number of seconds that SA OS/390 will wait before attempting to restart an application. This delay is provided to ensure that any shutdown processing has completed.</p> <p>5 is the default setting.</p>
AOFRPCWAIT	0 to n	<p>This is the number of seconds that SA OS/390 will wait for command responses from other systems in the sysplex.</p> <p>10 is the default setting.</p>
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.

## Global Variables to Enable Advanced Automation

Table 7. Global Variables to Enable Advanced Automation (CGLOBALS) (continued)

Variable	Value	Effect
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 <i>System Automation for OS/390 Planning and Installation</i> for more information.
AOFSETSTATEOVERRIDE	NO	Will set the default OVERRIDE parameter for the SETSTATE command to NO.  This is the default setting.
	FLG	Will set the default OVERRIDE parameter for the SETSTATE command to override all flag settings.
	TRG	Will set the default OVERRIDE parameter for the SETSTATE command to override all trigger settings.
	SVP	Will set the default OVERRIDE parameter for the SETSTATE command to override all service period settings.
	EVT	Will set the default OVERRIDE parameter for the SETSTATE command to override all event settings.
	ALL	Will set the default OVERRIDE parameter for the SETSTATE command to override all flag and trigger settings.
AOFSETSTATESCOPE	user defined valid value	allows you to override the predefined default for the SCOPE parameter of the SETSTATE command as documented in <i>System Automation for OS/390 Operator's Commands</i> .
AOFSETSTATESTART	user defined valid value	allows you to override the predefined default for the START parameter of the SETSTATE command as documented in <i>System Automation for OS/390 Operator's Commands</i> .
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	Will set the default OVERRIDE parameter for the INGREQ command to NO.  This is the default setting.
	FLG	Will set the default OVERRIDE parameter for the INGREQ command to override all flag settings.
	TRG	Will set the default OVERRIDE parameter for the INGREQ command to override all trigger settings.
	SVP	Will set the default OVERRIDE parameter for the INGREQ command to override all service period settings.
	EVT	Will set the default OVERRIDE parameter for the INGREQ command to override all event settings.
	ALL	Will set the default OVERRIDE parameter for the INGREQ command to override all flag and trigger settings.

## Global Variables to Enable Advanced Automation

Table 7. Global Variables to Enable Advanced Automation (CGLOBALS) (continued)

Variable	Value	Effect
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.
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.
	YES	
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.

## Global Variables to Enable Advanced Automation

Table 7. Global Variables to Enable Advanced Automation (CGLOBALS) (continued)

Variable	Value	Effect
INGREQ_RESTART	<b>NO</b>	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	<b>ONLY</b>	Sets the SCOPE parameter of the INGREQ command to the specified value.  ONLY is the default setting.
	ALL, CHILDREN	
INGREQ_TIMEOUT	<b>interval in minutes, MSG or CANCEL</b>	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	<b>NORM</b>	Sets the default startup/shutdown type (TYPE parameter) of the INGREQ command to the specified value.  NORM is the default setting.
	IMMED, FORCE	
INGREQ_VERIFY	<b>YES</b>	Sets the VERIFY parameter of the INGREQ command to the specified value.  YES is the default setting for autotasks.
	NO is the default setting for operator tasks.	
INGSET_VERIFY	<b>YES</b>	Sets the default VERIFY parameter of the INGSET command to the specified value.  YES is the default setting for autotasks.
	NO is the default setting for operator tasks.	
INGVOTE_VERIFY	<b>YES</b>	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.	

## Global Variables to Enable Advanced Automation

---

## 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 12 on page 122 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.

## Customizing the Status Display Facility (SDF)

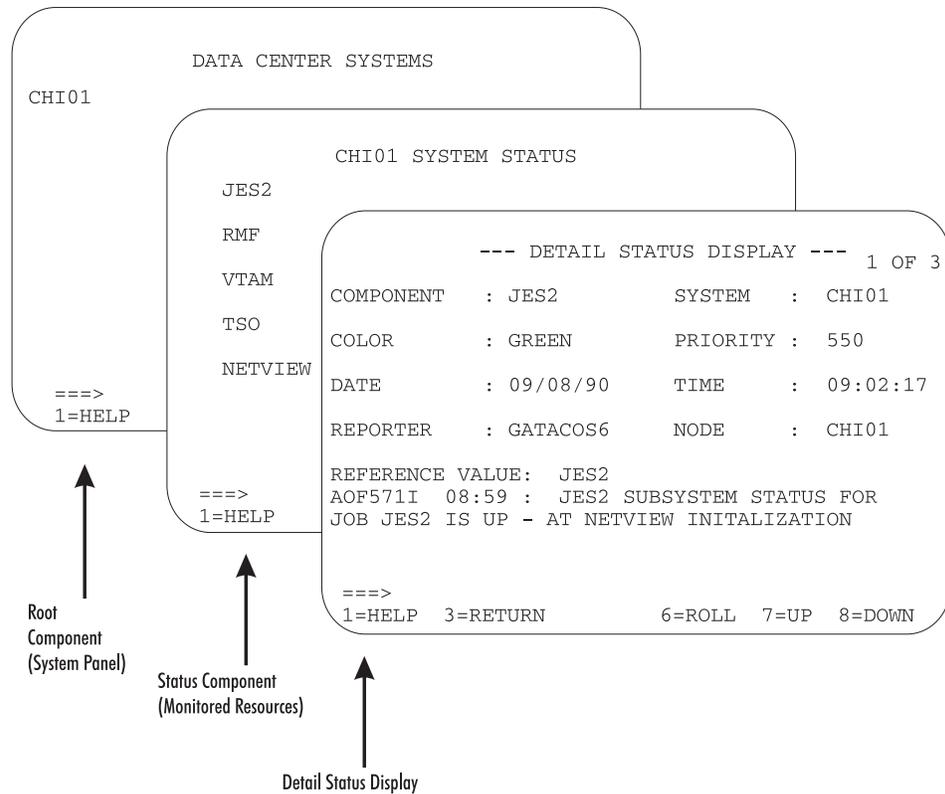


Figure 12. Example SDF Panels

### Root Component

The root component is typically an element appearing on the first screen displayed when SDF is started. In Figure 12, the CHI01 system is the root component.

### Status Component

Resources monitored by SDF are called *status components*. In Figure 12, 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 12 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

## Customizing the Status Display Facility (SDF)

- Priority, color, and highlighting associated with the status descriptor (see “How Status Descriptors Affect SDF” on page 125 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 122). 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 12 on page 122, 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 AOPTREE.

Figure 13 on page 124 shows an example SDF tree structure. Following the tree structure definition statements is a diagram showing how these statements result in a tree structure.

## Customizing the Status Display Facility (SDF)

```

1 SY1
2 SYSTEM
3 WTOR
3 APPLIC
4 AOFAPPL
5 AOFSSI
4 JES
4 VTAM
3 TSO
3 RMF
2 GATEWAY

```

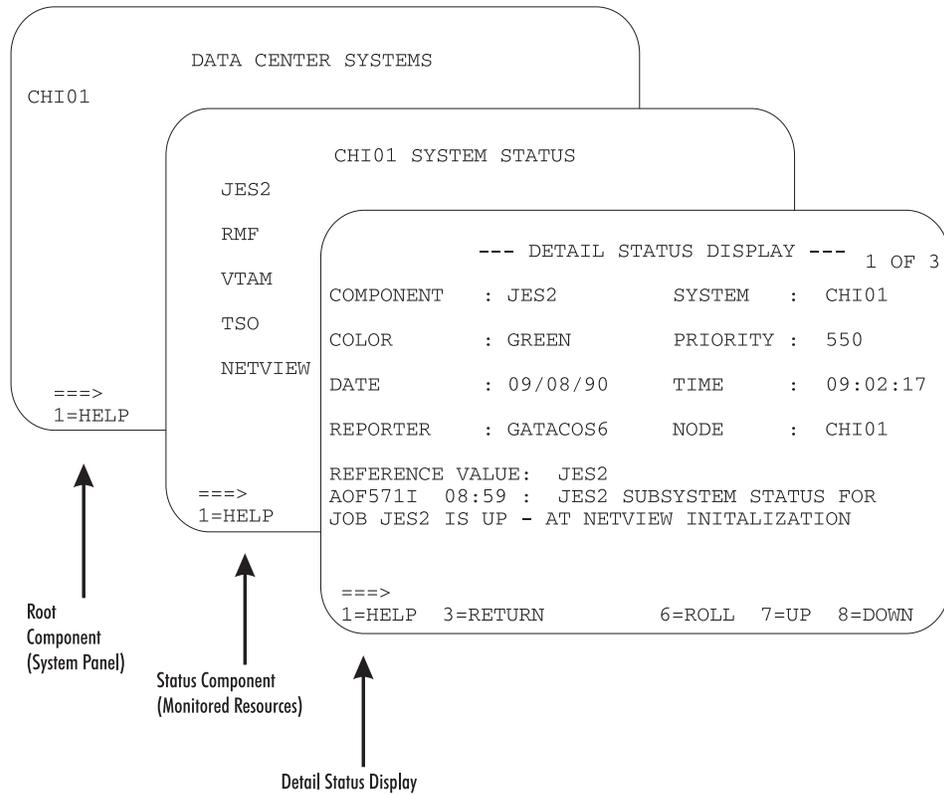


Figure 13. Example SDF Tree Structure

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 13, 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 132.

## 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 136.

### 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

## Customizing the Status Display Facility (SDF)

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 14 shows status descriptors currently generated for system SY1. The priority for each status descriptor is shown by a number.

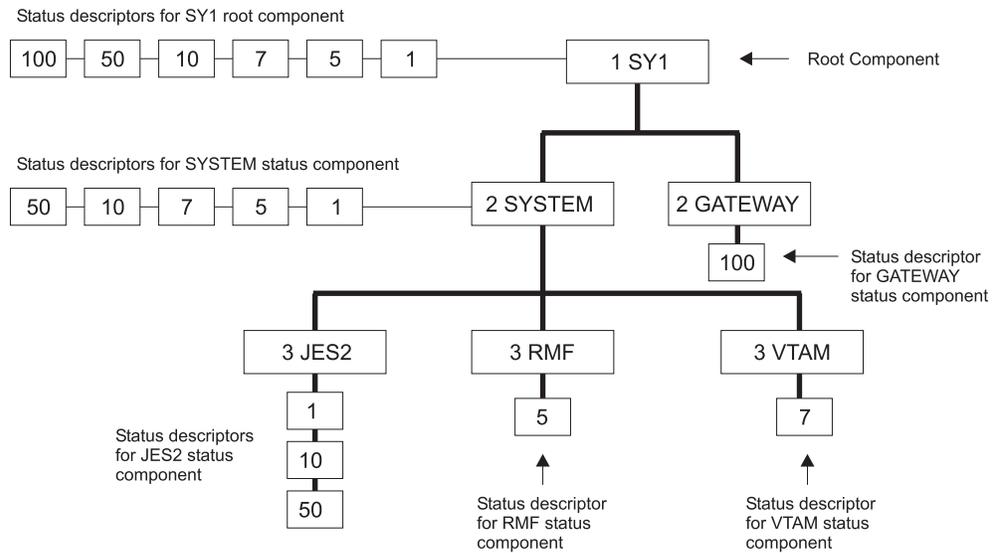


Figure 14. Status Descriptors Chained to Status Components

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.

## Customizing the Status Display Facility (SDF)

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 12 on page 122, 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 13 on page 124, 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 12 on page 122, 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

## Customizing the Status Display Facility (SDF)

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.

## Customizing the Status Display Facility (SDF)

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 13 on page 124, 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 133.

## 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.

## Customizing the Status Display Facility (SDF)

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 8. 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 <i>System Automation for OS/390 Programmer's Reference</i> . AOFINIT is in the NetView DSIPARM data set.
AOFTREE	Input file	Contains tree structures described in <i>System Automation for OS/390 Programmer's Reference</i> . 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 133. This member usually consists of a list of %INCLUDE statements referencing other members containing panel definitions. AOFPNLS is in the NetView DSIPARM data set.
<i>panel_name</i>	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.
<i>tree_name</i>	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 133 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.

## Customizing the Status Display Facility (SDF)

### 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 15 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 15. 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

## Customizing the Status Display Facility (SDF)

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 135 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 16 shows how an SDF panel looks when displayed:

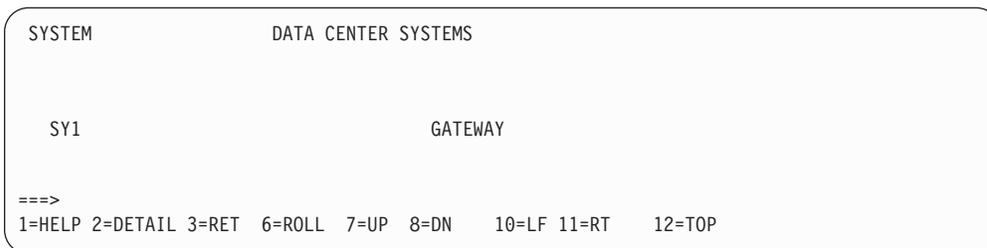


Figure 16. Example SDF Panel

Figure 17 shows the panel definition statements required to define the panel in Figure 16.

```
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 17. Example Panel Definition Entry

In Figure 17, 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 9 describes each statement in Figure 17:

Table 9. 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.

Table 9. Panel Definition Entry Description (continued)

Statement	Description and Example Value
TEXTFIELD(01,02,10,WHITE,NORMAL)	The text location statement defining constant panel fields. This field starts on line 01 in position 02 and ends in position 10. The color of the field is white and highlighting is normal.
TEXTTEXT(SYSTEM)	The text data statement specifying the actual data that goes in the text field just defined. This field contains the word SYSTEM.  TEXTFIELD and TEXTTEXT are always grouped in pairs.
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 <i>System Automation for OS/390 Programmer's Reference</i> 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 to be loaded. Member names in the list are delimited by a comma.
- The SDF %INCLUDE statement requires parentheses around the specified member or members.
- The target DSIPARM members may contain only complete panel definitions or additional %INCLUDE statements. Panel definitions must be contained within a single member, and therefore cannot be built using commonly defined segments.

### 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 the automation agents. The coordination, decision making and controlling functions are processed by each sysplex's *automation manager*.

The automation manager contains a model of all of the automated resources within the sysplex. The automation agents feed the automation manager with status information and perform the actions that the automation manager tells them to.

The automation manager provides *sysplex-wide* automation.

**Automation Manager Configuration.** The Automation Manager Configuration file (AMC) contains an image of the automated systems in a sysplex or of a 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*.

**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 SYSCONE 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 systems 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.** A console used by the operator to monitor and control a System/390 microprocessor cluster.

**Hardware Management Console Application (HWMCA).** A direct-manipulation object-oriented graphical user interface that provides single point of control and single system image for hardware elements. HWMCA provides customer grouping support, aggregated and real-time system status using colors, consolidated hardware messages support, consolidated operating system messages support, consolidated service support, and hardware commands targeted at a single system, multiple systems, or a customer group of systems.

**heartbeat.** In SA 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.

**Inter-System Control Facility (ISCF).** A program offering that provides remote operations support for 3090, 4381, and 308x processors. ISCF is a predecessor to SA OS/390. It runs as a command processor on the NetView program and works in conjunction with the ISCF/PC program.

**Inter-System Control Facility/PC (ISCF/PC).** A program offering that runs as an application on an IBM personal computer and works in conjunction with ISCF.

**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.

**ISCF.** Inter-System Control Facility.

**ISCF/PC.** Inter-System Control Facility/Personal Computer.

**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 communicate 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)

**OS/2.** A software operating system for personal computers (workstations). OS/2 supports sophisticated business applications and gives you multiple simultaneous host sessions, DOS sessions, Windows sessions, and OS/2 sessions with a graphic interface that lets you select resources by clicking on an icon with a mouse.

**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 TPF). 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.

**OS/2.** The IBM Operating System/2 program.

**OS/2 program.** The IBM Operating System/2\* program. SA OS/390 uses the intersystem communication and emulation services of the Communication Manager.

**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.

**Personal System/2\* (PS/2) computer.** A family of microcomputers manufactured by IBM Corporation.

**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.

**PS/2 computer.** In SA OS/390, an IBM PS/2 computer that runs the OS/2 program and that contains a communication adapter. The PS/2 computer, which is connected to one or more target systems, contains the distributed feature code of SA OS/390. An SA OS/390 workstation may be either an IBM PS/2 computer or an IBM PS/55 computer, unless otherwise stated.

**PS/2 LU name.** In SA OS/390, the focal-point SNA logical unit name of the workstation.

**PS/2 name.** In SA OS/390, the logical name of the workstation.

**PS/2 ports.** On a PS/2 workstation, a port is a cable connector on a 3270 or ASCII logic card that attaches it to a target system. The port is associated with the screen handler for the PS/2 and identified by a port letter.

**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-based processor.** See *screen-oriented processor*.

**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.

**screen-oriented processor.** A central processor complex that interacts with human operators through line-mode display consoles. For System/390 and System/370 processors, the display consoles use either an IBM 3270 data stream or an ASCII data stream. Contrast with *OCF-based processor*.

**SDF.** Status display facility.

**SDLCL.** Synchronous data link control.

**SDFS.** 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.

**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 (WTOs) 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.

**TPF.** Transaction processing facility.

**transaction processing facility (TPF).** A high-availability, high-performance system, designed to support real-time, transaction-driven applications (such as an airline reservation system). The specialized architecture of TPF is intended to optimize system efficiency, reliability, and responsiveness for data communication and database processing.

TPF provides real-time inquiry and update to a large, centralized database, where message length is relatively short in both directions, and response time is generally less than three seconds. Formerly known as the Airline Control Program/Transaction Processing Facility (ACP/TPF).

**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

**3090/ACA/ISCF.** 3090 Automated Console Application for Inter-System Control Facility.

**3090/Automated Console Application/ISCF (3090/ACA/ISCF).** A program offering that runs as a NetView application in conjunction with ISCF. It provides some built-in automation and an enhanced user interface. 3090/ACA/ISCF is limited to 3090 OS/390 targets.

**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

## A

- abnormal end messages 28
- ACTIVE messages 27
- ACTIVMSG 27, 33
- adding a message to automation 39
- adding an application to automation 31
- adding an entry to the NetView message automation table 41
- advanced automation options
  - exits 85
  - external global variables 107, 113
- AOCMSG call 61
- AOCMSG generic routine 55
- AOCQRY common routine
  - automation availability 54
  - message automation 70
- AOCTRACE
  - use in testing 64
  - use in traces 67
- AOCUPDT common routine
  - and the AOFEXSTA exit 85
  - to update status information 55
- AOF.0DEBUG 107
- AOF.0TRACE 107
- AOF\_ASSIGN\_JOBNAME 107, 113
- AOF\_EMCS\_AUTOTASK\_ASSIGNMENT 107, 113
- AOF\_EMCS\_CN\_ASSIGNMENT 108, 113
- AOF\_GATOPER\_PW\_CHANGE 108, 114
- AOF\_INIT\_MCSFLAG 108, 115
- AOF\_INIT\_ROUTCDE 108, 115
- AOF\_INIT\_SYSCONID 108, 115
- AOF\_NETWORK\_DOMAIN\_ID 108
- AOF\_PRODLVL 109
- AOF\_SET\_AVM\_RESTART\_EXIT 110, 117
- AOF3WTIME 110, 118
- AOFACFINIT 107, 113
- AOFADMON 49
- AOFAJMON 49
- AOFAOCCLONE 107
- AOFAPMON 49
- AOFARMPPI 14
- AOFATMON 49
- AOFCONMASK 107, 114
- AOFCONFIRM global variable 92
- AOFPCSM 49
- AOFCTLOPT 107, 114
- AOFDEBUG 107
- AOFDEBUG global variable 68
- AOFDEFAC 5
- AOFDEFAULT\_TARGET 107, 114
- AOFDOM 5
- AOFEXC00 exit 87
- AOFEXC01 exit 87
- AOFEXDEF exit 84, 113
- AOFEXI01 exit 85
- AOFEXI02 exit 85
- AOFEXINT exit 85, 93, 110, 116
- AOFEXPLAIN\_USER 108, 114
- AOFEXSTA exit 85
- AOFGMFHSWAIT 14
- AOFHOLDA 6
- AOFHOLDD 6
- AOFHOLDE 6
- AOFHOLDI 5
- AOFHOLDW 7
- AOFINITIALSTARTTYP 108
- AOFINITREPLY 109, 115
- AOFJES3JOB 7
- AOFJES3PREFIX 109
- AOFLOCALHOLD 109, 115
- AOFLOPAUTOx 7
- AOFLOPMSU 13
- AOFMAPP 15
- AOFMGMFH 16
- AOFMINORCHK 109, 115
- AOFMIO 16
- AOFMJES2 16
- AOFMJES3 16
- AOFMMVS 17
- AOFMNETV 17
- AOFMOVOP 109, 115
- AOFMPOPS 17
- AOFMRMF 18
- AOFMRODM 18
- AOFMSG00 4
- AOFMSG01 4, 24
- AOFMSG03 4
- AOFMSGBL 20
- AOFMSGGN 20
- AOFMSGGN sample 27
- AOFMSGGW 18
- AOFMSGHI 20
- AOFMSGMS 20
- AOFMSGNM 21
- AOFMSGPA 19
- AOFMSGPM 19
- AOFMSGST 19
- AOFMSGSY 4, 18
- AOFMSGTS 19
- AOFMSGXC 21
- AOFMTSO 21
- AOFMVLLA 22
- AOFMVTAM 22
- AOFNORMA 8
- AOFNORMD 8
- AOFNORME 8
- AOFNORMI 8
- AOFNORMW 9
- AOFOPAUTO1 9
- AOFOPAUTO2 10
- AOFOPBASEOPER 10
- AOFOPGATOPER 13
- AOFOPGSSOPER 12
- AOFOPJESOPER 11
- AOFOPMONOPER 11
- AOFOPMSGOPER 10
- AOFOPMSU 13
- AOFOPNETOPER 11
- AOFOPRECOPER 11
- AOFOPSHUTOPER 12
- AOFOPSYSOPER 10
- AOFOPTOPOMGR 14
- AOFOPWTORS 12
- AOFPAUSE 109, 116
- AOFQUICKWTOR 109, 116
- AOFRELOADOPT 109, 116
- AOFRESTARTALWAYS 109, 116
- AOFRESTARTDELAY 109, 116
- AOFRPCWAIT 109, 116
- AOFSENDGMFHSREQUEST 110
- AOFSEXINT 110, 116
- AOFSETSTATEOVERRIDE 110, 117
- AOFSETSTATESCOPE 110, 117
- AOFSETSTATESTART 110, 117
- AOFSHUTCHK 110
- AOFSHUTDELAY 110, 117
- AOFSHUTOVERRIDE 110, 117
- AOFSHUTSCOPE 110
- AOFSIRTASK 13
- AOFSPOOLFULLCMD 110, 118
- AOFSPOOLSHORTCMD 110, 118
- AOFSUBSYS 110
- AOFSYS 5
- AOFSYSNAME 110
- AOFSYSTEM 111
- AOFTDDF task 65
- AOFTECMODE 15
- AOFTECPPI 15
- AOFTECTASK 14
- AOFTECTASKQ 14
- AOFUSSWAIT 118
- AOFUXMON 49
- AOFVTAMJOB 9
  - application
    - adding to automation 31
  - application messages, entries in MPF list 33
  - application monitor status 49
  - application monitoring 49
  - ASCB chaining
    - and global variables 113
  - ASFUSER command 70
  - assist mode
    - for testing automation procedures 65
    - monitoring automation with interactive 66
    - overview 64
  - automating processor operations controlled resources 73
  - automation
    - adding an application 31
    - advanced functions 113
    - assigning a message 47
    - extending 51
    - response to messages 39
  - automation configuration 32
  - automation control file 32
    - defining SDF 136
    - reload action exit 92

- automation control file (*continued*)
  - reload permission exit 92
- automation flag
  - global 26
- automation flags 25
  - checking 26
  - example for using 25
  - for minor resources 25
  - with individual messages 25
  - with status changes 25
- automation manager configuration file 32
- automation manager global automation flag 26
- automation procedure
  - example 62
- automation procedures
  - calling 52
  - creating 51
  - description 51
  - developing messages 60
  - external code 56
  - initializing 53
  - installing 64
  - making generic 59
  - structure 52
  - testing and debugging 64
  - using AOCTRACE 67
  - writing your own 51
- automation programmer xiii
- automation status file 139
  - coding your own information 69
  - using commands 55
- automation table
  - fragments 15
  - generic entries 27
  - structure 3
- automation table statements
  - user-written 22
- automation tables
  - abend messages 28
  - AOFMSGSY 4
  - fragments 4, 24
  - integrating 22
  - master automation tables 4
  - message addition 39
  - samples 3
  - status transition messages 27

## B

- BASEOPER 116
- black box 23
- BLDVIEWS 37

## C

- casca des 9
- CDEMATCH common routine 70
- CHKTHRES common routine 55
- clone ID
  - Automatic Restart Manager 121
- code matching 40
- command processing 40
- common global variables 56, 107
- common routines 51

- configuration data
  - cross-reference 99
  - maintaining processor operations PC 97
  - PS/2, keeping synchronized 98
- console operation
  - operator console 56
  - system console 56
- CPCB (command processor control block) 141
- cross-reference, PS/2 configuration data 99
- customization dialog exits 93
  - invocation 95
- customize automation
  - for processor operations 56
  - for system operations 54
- customizing SDF 121

## D

- defining
  - SDF in automation control file 136
- DISPASST 65
- DSICMD member 64
- DSIPARM
  - description 142
- DSIPARM data set 64

## E

- editing the profile data set 97
- element names
  - in Automatic Restart Manager 121
- environmental setup exits 88
- error codes 55
- exits 92
  - AOFEXC00 87
  - AOFEXC01 87
  - AOFEXDEF 84
  - AOFEXI01 85
  - AOFEXI02 85
  - AOFEXINT 85, 93
  - AOFEXSTA 85
  - BUILDF processing 93
  - CONVERT processing 95
  - COPY processing 94
  - customization dialog exits 93
  - DELETE processing 94
  - environmental setup exits 88
  - flag exits 89
  - INGEAXIT 89
  - INGEX01 93
  - INGEX02 93
  - INGEX03 94
  - INGEX04 94
  - INGEX05 94
  - INGEX06 94
  - INGEX07 95
  - INGEX08 95
  - pseudo-exits 92
  - sample automation flag exits 92
  - static exits 84
  - status change commands 87

- exits (*continued*)
  - subsystem up at initialization commands 93
  - testing 93
- EXPLAIN 108, 114
- extended automation flags 25
- extending
  - automation 51
- external common global variables 107
- EXTSTART status 121

## F

- file manager commands 55
- FINAL TERMINATION messages 28
- flag exits 89
- fragments of automation table 15

## G

- generic
  - automation 27, 113
  - routines 51
- generic automation procedures 59
- generic routines 51
- global automation flag 26
- graphic workstation description 147

## H

- HWMCA
  - description 143

## I

- IDENT 70
- IGNORE WTOR priority 32
- IMPORTANT WTOR priority 32
- INCLUDE statement 135
- INGAUTO\_INTERVAL 111, 118
- INGDLG 95
- INGEAXIT exit 89
- INGEIO 144
- INGEX01 93
- INGEX02 93
- INGEX03 94
- INGEX04 94
- INGEX05 94
- INGEX06 94
- INGEX07 95
- INGEX08 95
- INGMTEC 22
- INGREQ\_ORIGINATOR 111, 118
- INGREQ\_OVERRIDE 111, 118
- INGREQ\_PRECHECK 111, 118
- INGREQ\_PRI 111, 118
- INGREQ\_REMOVE 111, 118
- INGREQ\_RESTART 111, 119
- INGREQ\_SCOPE 111, 119
- INGREQ\_TIMEOUT 111, 119
- INGREQ\_TYPE 111, 119
- INGREQ\_VERIFY 111, 119
- INGSAMPL 97
- INGSET\_VERIFY 112, 119
- INGVOTE\_VERIFY 112, 119

- initialization processing
  - AOFSERXINT 110, 116
- installing automation procedures 64
- ISQEXEC command 43, 58
- ISQMTSYS 49
- ISQOVRD 44
- ISQOVRD command 59
- ISQTCC command 59
- ISQXLOC command 59
- ISQXMON command 43
- ISQXUNL command 59
- ISSUECMD 25, 40
- ISSUEREP 25, 40
- IWTOR 32
- IXC102A message automation 77

## K

- keeping PS/2 configuration data
  - synchronized 98

## L

- Linux console connection to NetView 82

## M

- maintaining processor operations PC
  - configuration data 97
- major resources 25, 92
- master automation tables 4
  - multiple 23
- message
  - ISQ900I 43
  - ISQ901I 43
  - types 3
- message automation 76
- message automation for IXC102A 77
- message automation for Linux console
  - messages 82
- message automation for processor
  - operations resources 73
- message presentation 5
- message processing
  - user specific 41
- message processing facility list
  - adding application messages 33
- message testing 46
- messages
  - assigning to automation
    - environment 47
  - automating response to 39
  - classifications 3
  - defining as minor resources 25
  - forwarding 43
  - status transition 27
  - suppressing 47
  - testing 43, 46
- messages, entries in MPF list 33
- minor resources
  - and INGAUTO 25
  - and task globals 92
  - AOFMINORCHK 115
  - defining message and status 25
  - resource name 91
- monitor routine 49

- monitor routine (*continued*)
  - writing own 49
- monitoring
  - automation with interactive assist
    - mode 66
  - enable an application 37
- monitoring applications 49
- MPF list 37
  - adding an entry 47
  - adding application messages 33
- MVS Automatic Restart Manager
  - global variables 121
- MVSESA.RELOAD.ACTION minor
  - resource 92
- MVSESA.RELOAD.CONFIRM flag 92
- MVSESA.RELOAD.CONFIRM minor
  - resource 92

## N

- Netfinity 56
- NetView
  - generic automation table entries 27
  - testing and debugging facilities 69
- NetView automation table
  - adding entry calling ACTIVMSG 33
  - adding entry calling
    - ACTIVMSGUP=YES 33
  - adding entry calling TERMMSG 34
  - adding entry calling TERMMSG
    - FINAL=YES 34
  - adding SDF entries 35
  - ISQEXEC 43, 58
  - ISQOVRD 59
  - ISQTCC 59
  - ISQXLOC 59
  - ISQXMON 43
  - ISQXUNL 59
- merging entries 46
- performance consideration 41
- production 46
- reloading tables 37
- sample entry 44
- NetView message automation table 39
  - adding an entry 41
  - generic entries 27
- NMC focal point system 142
- NMC workstation 37
- NONSNA 37
- NORMAL WTOR priority 32
- notifications 55
- NWTOR 32

## O

- operator xiii
- operator cascades 9
- outstanding reply processing 31

## P

- PolyJes environment 5
- PRI WTOR type 32
- processor operations
  - dedicated personal computer 150
  - PS/2 profile 107

- processor operations (*continued*)
  - sample profile INGSAMPL 97
- processor operations command
  - messages 44
- processor operations commands 60
- processor operations controlled resources
  - automating 73
- processor operations PC
  - xix
  - profiles for processor operations 97
- processor operations PC configuration
  - data
    - maintaining 97
- processor operations PS/2
  - xix
- processor operations resource 73
  - message automation 73
- ProcOps PC
  - xix
- product automation 150
- profile
  - data set 97
  - editing 97
  - sample SA OS/390 processor
    - operations
      - INGSAMPL 97
      - PS/2 97
- programming recommendations 70
- proxy resource 73
- proxy resources
  - automation for 74
- PS/2
  - xix
- PS/2 computer
  - configuration data cross-reference 99
- PS/2 configuration data
  - keeping synchronized 98
- pseudo-exits 92

## R

- reload action exit 92
- RELOAD.ACTION flag 92
- RELOAD.CONFIRM flag 92
- reload permission exit 92
- reloading NetView automation table 37
- reply processing 40
  - outstanding 31
- REXX PARSE 70
- REXX trace type 68
- routines
  - common 51
  - generic 51
- RWTOR 32

## S

- SA OS/390
  - commands ISQXIPM and
    - ISQCMMT 57
- sample
  - automation tables 3
  - processor operations profile
    - INGSAMPL 97
  - SA OS/390 processor operations
    - profile PS/2 97
- SDF
  - and specific problems 128
  - components 130
  - customizing 121

SDF (*continued*)

- customizing initialization
  - parameters 136
- defining hierarchy 132
- defining in automation control
  - file 136
- defining in customization dialog 136
- defining panels 133
- definition process 131
- for multiple systems 129
- how it works 121
- panels
  - definition 128, 133
  - types 121
- starting and stopping 131
- status descriptors 122
- tree structures 123

SDF entries 35

SDF panels 36

SDF tree structure 35

SEC WTOR type 32

serialize command processing 57

SETASST 65

status
 

- defining as minor resources 25

status change commands 87

status descriptors 125
 

- chaining to status components 125
- propagating 127

status file 139

status information 55

status transition messages 27

SUBSAPPL 70

SUBSJOB 70

SUBSTYPE 70

subsystem
 

- adding to automation 31
- up at initialization commands 93

suppressing messages 47

synchronizing PS/2 configuration
 

- data 98

system operations control files 47
 

- automation control file 32
- automation manager configuration file 32

system programmer xiii

## T

task global variables 56

TERMMSG 28, 34

testing
 

- messages 46

testing and debugging automation
 

- procedures 64

testing and debugging facilities 69

testing exits 93

## U

UCB (unit control block) 155

UNUSUAL WTOR priority 32

user exits 83
 

- static exits 84

user groups
 

- automation programmer xiii

user groups (*continued*)
 

- operator xiii
- system programmer xiii

user specific message processing 41

UWTOR 32

## V

VTAM
 

- and assist mode 65

## W

WAITTIME 112

WTOR
 

- priority 31
- type 31

## X

XDOMTIME 112

---

# Readers' Comments — We'd Like to Hear from You

System Automation for OS/390  
Customizing and Programming  
Version 2 Release 1

Publication No. SC33-7035-02

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>				

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>				
Complete	<input type="checkbox"/>				
Easy to find	<input type="checkbox"/>				
Easy to understand	<input type="checkbox"/>				
Well organized	<input type="checkbox"/>				
Applicable to your tasks	<input type="checkbox"/>				

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.



Fold and Tape

Please do not staple

Fold and Tape



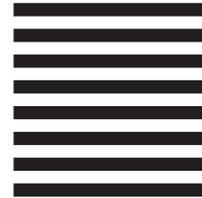
NO POSTAGE  
NECESSARY  
IF MAILED IN THE  
UNITED STATES

# 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





Program Number: 5645-006



Printed in the United States of America  
on recycled paper containing 10%  
recovered post-consumer fiber.

SC33-7035-02



Spine information:



Customizing and Programming  
System Automation for OS/390 V2R1.0  
Customizing and Programming

Version 2 Release 1

SC33-7035-02