

---

---

# Unicenter

## Automation Services Administrator Guide

12th edition – update 1

P01-172



**Computer Associates**  
The Software That Manages eBusiness



---

<b>Edition</b>	<b>Publication Number</b>	<b>Product Version</b>	<b>Min. MS Level</b>	<b>Publish Date</b>
12th Edition—Update 1	P01-172	5.0	5.0	December 2001

---

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

This documentation may not be copied, transferred, reproduced, disclosed or duplicated, in whole or in part, without the prior written consent of CA. This documentation is proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of this documentation for their own internal use, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the confidentiality provisions of the license for the software are permitted to have access to such copies.

This right to print copies is limited to the period during which the license for the product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to return to CA the reproduced copies or to certify to CA that same have been destroyed.

To the extent permitted by applicable law, CA provides this documentation "as is" without warranty of any kind, including without limitation, any implied warranties of merchantability, fitness for a particular purpose or noninfringement. In no event will CA be liable to the end user or any third party for any loss or damage, direct or indirect, from the use of this documentation, including without limitation, lost profits, business interruption, goodwill, or lost data, even if CA is expressly advised of such loss or damage.

The use of any product referenced in this documentation and this documentation is governed by the end user's applicable license agreement.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227-7013(c)(1)(ii) or applicable successor provisions.

© 2001 Computer Associates International, Inc.

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

---

# Table of Contents

## Part I      **Implementing and Administering Automation Services**

<b>Chapter 1</b>	<b>Introduction .....</b>	<b>1-1</b>
	What Automation Services Offers .....	1-2
	Features .....	1-3
	AutoAssist.....	1-3
	User Management.....	1-3
	Multisystem Support.....	1-4
	Product Administration.....	1-5
	System Programming.....	1-5
	Automation Services Application Program Interface.....	1-6
	The DD SUBSYS Advanced Facility.....	1-6
	Management Services .....	1-6
	About this Guide .....	1-7
	What You Should Already Know .....	1-7
	Notational Conventions.....	1-7
	Variables Representing Data Set Names .....	1-7
	Product Specific Information.....	1-8

<b>Chapter 2</b>	<b>Initializing, Implementing, and Customizing a Region .....</b>	<b>2-1</b>
	Initialization and Customization Services .....	2-2
	What are Parameter Groups? .....	2-2
	Initializing a Region for the First Time .....	2-3
	Implementing a SOLVE:Problem Automatic Problem Recording Environment .....	2-3
	Task 1—Copying the \$RMPB07S NCL Procedure .....	2-3
	Task 2—Defining Users in the SOLVE:Problem Region.....	2-3
	Task 3—Defining Links.....	2-4
	Implementing the Alert Monitor Problem Ticket Interface.....	2-4
	Defining a Problem Ticket Interface.....	2-5
	Defining an E-mail Problem Ticket Interface.....	2-5
	Defining a Custom Problem Ticket Interface .....	2-7
	Setting up the Problem Ticket Data Definition.....	2-8
	Considerations.....	2-9
	Dealing With Initialization Failures .....	2-10
	Using the System Initialization In Progress Dialog .....	2-11
	Setting Up Checkpoint Restart of a System Image .....	2-12
	Customizing External Applications Access.....	2-13
	Navigating to an External Application.....	2-14
	Generating an INI Procedure .....	2-16
	Using an INI Procedure to Propagate Customized Parameter Group Records to Other Regions .....	2-17
<b>Chapter 3</b>	<b>Implementing Security .....</b>	<b>3-1</b>
	About Security .....	3-2
	Controlling Signon Access.....	3-2
	UAMS .....	3-3
	External Security Packages.....	3-3
	Controlling Access to Functions and Resources .....	3-5
	Network Partitioning Facility (NPF) .....	3-5
	External Security Options .....	3-7
	Generating Background User IDs.....	3-9
	Background User Considerations for Existing UAMS File.....	3-10

Customizing Parameters That Affect Security .....	3-11
Command Replacement.....	3-11
Synchronizing Updates Across Linked Regions .....	3-11
Troubleshooting.....	3-13
User Profiles—Levels of Access and Privilege .....	3-14
Defining Users by Using User Profiles .....	3-15
Specifying a User’s Details.....	3-16
Customizing a User’s Resource Monitor Display .....	3-17
Customizing a User’s Graphical Monitor .....	3-19
Customizing a User’s Alert Monitor .....	3-20
Customizing a User’s Message Monitor Profile.....	3-20
Customizing a User’s Consolidated Console.....	3-21
Customizing a User’s SNA Network Summary Display.....	3-23
Maintaining User Profiles .....	3-23
Updating User Profiles.....	3-24
Deleting a User Profile Definition .....	3-24
Defining a User ID With Special Security Requirements.....	3-25
Defining Users For An External Security Package .....	3-25
Authorizing VM Programmable Operator Interface Users .....	3-26
Controlling Access to Functions and Resources by Using NPF .....	3-26
Controlling Access to Menu Options.....	3-27
Controlling Access to the Knowledge Base .....	3-27
Controlling the Type of Access .....	3-28
Controlling Access by System Images .....	3-28
Controlling Access by Classes .....	3-28
Controlling Access by Resources .....	3-29
Controlling Access to System Images .....	3-29
Controlling Access to Commands .....	3-29
Controlling access to Automation Services Commands.....	3-30
Controlling Access to System Commands .....	3-30
Controlling Access to Management Services (MS) Commands .....	3-30
Controlling Access to Commands Issued Against Systems .....	3-31
Controlling Access to Commands Issued Against Resources..	3-31
Controlling Access to Management Services Commands from the Message Monitor .....	3-32
Controlling Access to Initialization and Customization Parameter Groups .....	3-33
Changing an NPF Table.....	3-33
Implementing Security Through an External Security Package .....	3-34

	Controlling Access to Functions and Resources by Using an External Security Package .....	3-34
	Controlling Access to Menu Options .....	3-35
	Controlling Access to the Knowledge Base .....	3-35
	Controlling Access to System Images.....	3-39
	Controlling Access to Automation Services Commands .....	3-39
	Controlling Access to System Commands .....	3-41
	Controlling Access to Management Services Commands .....	3-41
	Controlling Access to Initialization and Customization Parameter Groups.....	3-42
	Securing Data Set Members .....	3-42
<b>Chapter 4</b>	<b>Starting and Stopping a Region.....</b>	<b>4-1</b>
	Starting a Region .....	4-2
	Responding to WTOR Confirmation Message .....	4-2
	Continue Startup With no Change .....	4-2
	Continue Startup With Changes.....	4-3
	Shutting Down a Region.....	4-4
	SHUTDOWN Command .....	4-4
	FSTOP Command .....	4-4
<b>Chapter 5</b>	<b>Administering a Multisystem Environment .....</b>	<b>5-1</b>
	Multisystem Operation .....	5-2
	Links in a Multisystem Environment.....	5-4
	Multisystem Support in Sysplex .....	5-5
	Multisystem Implementation Considerations .....	5-6
	Establishing Regions .....	5-6
	Types of Data.....	5-7
	Linking Regions and Synchronizing Databases .....	5-8
	Performing the Link and Synchronize Procedure .....	5-10
	Monitoring the Synchronization Procedure .....	5-14
	Background User Considerations.....	5-15
	Knowledge Base Synchronization Maintenance.....	5-16
	Keeping Track of Linked Regions.....	5-16
	Unlinking Regions .....	5-19

	Transmitting Records .....	5-20
	Transmission Modes .....	5-20
	Transmission Procedure.....	5-21
	Developing and Testing a System Image in Isolation .....	5-24
<b>Chapter 6</b>	<b>Maintaining Automation Services .....</b>	<b>6-1</b>
	Changing the Activity Log .....	6-2
	Database Maintenance.....	6-3
	Unlinked Non-production Regions .....	6-3
	Unlinked Production Regions .....	6-4
	Linked Regions .....	6-5
	Backups to Tape When Regions Cannot Be Shut Down.....	6-5
<b>Chapter 7</b>	<b>Customizing the Display Attribute Tables .....</b>	<b>7-1</b>
	What Is a Display Attribute Table?.....	7-2
	Editing a Display Attribute Table .....	7-2
	The Logical State Attributes Table .....	7-3
	Logical State Attributes and SNA Network Summary Display ....	7-4
	The Automated Mode Attributes Table.....	7-4
	Setting the Logical States for the Automated Mode.....	7-4
	The Manual Mode Attributes Table .....	7-5
	Setting the Logical States for the Manual Mode .....	7-5
	The SNA Resource Logical State Normalization Table .....	7-6
	Setting the Logical States .....	7-6
<b>Chapter 8</b>	<b>Customizing Display Formats.....</b>	<b>8-1</b>
	What Is a Status Monitor Display Format? .....	8-2
	Defining Status Monitor Display Formats .....	8-3
	Creating a Status Monitor Display Format .....	8-4
	Specifying the Status Monitor Display Format .....	8-6
	Specifying a Multiscreen Display Format .....	8-9
	What Is an Alert Monitor Display Format?.....	8-10
	Defining Alert Monitor Display Formats .....	8-10
	Creating an Alert Monitor Display Format.....	8-11
	Specifying the Alert Monitor Display Format.....	8-12

<b>Chapter 9</b>	<b>Maintaining Prompt Lists .....</b>	<b>9-1</b>
	Editing a Prompt List.....	9-2
	Adding a Value to a List .....	9-2
	Using Variables.....	9-3
	Using Less-than Signs (<) to Represent a Left-justified, Fixed-length Variable Field Value.....	9-3
	Using Greater-than Signs (>) to Represent a Right-justified, Fixed-length Variable Field Value.....	9-4
	Maintaining Prompt List Definitions .....	9-4
	Maintaining List Entries .....	9-5
<b>Chapter 10</b>	<b>Defining Macros .....</b>	<b>10-1</b>
	What Is a Macro?.....	10-2
	Writing NCL Procedures to Be Used as Macros .....	10-2
	Customizing the \$RMMC00S Macro Template .....	10-2
	Customizing the \$RMMC00D Macro Template.....	10-3
	Registering and Maintaining the Macros.....	10-3
	Accessing the Macro Definitions .....	10-3
	The Macro Definition Panel.....	10-4
	Adding and Maintaining Macro Definitions .....	10-4
<b>Chapter 11</b>	<b>Defining Commands .....</b>	<b>11-1</b>
	What Is an Automation Services Command?.....	11-2
	Registering and Maintaining Commands .....	11-3
	Accessing Command Definitions.....	11-3
	The Command Details Panel.....	11-4
	The Command Prompting, Confirmation and Validation Panel....	11-5
	Adding and Maintaining Command Definitions.....	11-6
	Defining Your Own Commands .....	11-6
	Coding Your NCL Procedure.....	11-7
	Commands That Execute on Remote Systems .....	11-7
	Variables Available to a Command NCL Procedure .....	11-7

## Part II Reference Material

<b>Appendix A</b>	<b>Security Settings.....</b>	<b>A-1</b>
	Security Settings for \$RMADMIN .....	A-2
	Security Settings for \$RMOPER.....	A-3
	Security Settings for \$RMNOPER.....	A-5
	Security Settings for \$RMMON.....	A-6
	Security Settings for \$RMBUSER .....	A-7
<b>Appendix B</b>	<b>Parameter Groups.....</b>	<b>B-1</b>
	List of Parameter Groups .....	B-2
<b>Appendix C</b>	<b>Consoles.....</b>	<b>C-1</b>
	JES, OP1, OP2, and Pseudo Consoles.....	C-2
	Extended MCS Consoles .....	C-2
	Migration IDs.....	C-3
	Console Management .....	C-4
	Console Pool Management .....	C-4
	Console Acquisition .....	C-4
	Console Release.....	C-4
	Specifying the Number of Consoles.....	C-5
	Considerations .....	C-5
<b>Appendix D</b>	<b>Non-VTAM Terminal Support.....</b>	<b>D-1</b>
	Implementing Non-VTAM Terminal Support by Using a Local Terminal .....	D-2
	Sysplex Support .....	D-2
	Enabling Terminal Support in SSI.....	D-3
	Specifying Accessible Regions.....	D-3
	Enabling Automatic Logon to a Region .....	D-3
	TERMACCESS Parameter.....	D-4
	Attaching and Detaching Terminals .....	D-5
	Defining Terminal Names .....	D-5
	Activating the SSI.....	D-5
	Displaying Attached Terminals .....	D-5
	Controlling Non-VTAM Terminals Through Customization Parameters.....	D-6

	Remote Device System (RDS) Considerations.....	D-6
	Using a Non-VTAM Terminal .....	D-7
	Accessing a Defined Region .....	D-8
	Using the SYSREQ Key .....	D-8
	Terminating a Session .....	D-9
	Session Status.....	D-9
	Commands .....	D-10
	Implementing Non-VTAM Terminal Support by Using Telnet.....	D-11
	Connecting to the Automation Services Region by Using Telnet.....	D-11
<b>Appendix E</b>	<b>SMF Record Format .....</b>	<b>E-1</b>
	About Automation Services SMF Records.....	E-2
	SMF Header Format .....	E-2
	EventView SMF Record Format .....	E-3
	ResourceView and ServiceView Record Format .....	E-4
	User-defined Record Format .....	E-4
<b>Appendix F</b>	<b>Standardization of NCL Exits .....</b>	<b>F-1</b>
	What Is an NCL Exit Procedure? .....	F-2
	Standardized Structure.....	F-2
	Introducing the Exit.....	F-2
	Parameter Processing .....	F-3
	Parameters Passed as NCL Variables .....	F-3
	Parameters Passed Directly to the Exit .....	F-3
	Main Processing Section .....	F-5
	Exiting Back to the Caller .....	F-5

## Appendix G Using the Application Program Interface ..... G-1

Application Programming Interface Procedures .....	G-2
About the \$RMCALL API .....	G-3
About the \$RMDBAPI API .....	G-3
About the \$RMEVENT API .....	G-4
About the \$RMSTSET API .....	G-5
About the \$RECALL API .....	G-5
About the \$REDBAPI API .....	G-6
\$RMCALL ACTION=COMMAND .....	G-7
Examples .....	G-10
Supplied Commands That Require Parameters .....	G-11
GLOBAL Command .....	G-11
LOAD Command .....	G-11
\$RMCALL ACTION=DBGET .....	G-12
Example .....	G-13
\$RMCALL ACTION=PURGE .....	G-14
\$RMCALL ACTION=STGET .....	G-16
Example .....	G-17
\$RMDBAPI SERVICE=CREATE .....	G-18
Example .....	G-21
ResourceView Definition Field Names .....	G-21
System Image Fields .....	G-22
Resource Fields .....	G-23
\$RMDBAPI SERVICE=DELETE .....	G-32
Example .....	G-34
\$RMDBAPI SERVICE=GET .....	G-35
Example .....	G-37
\$RMDBAPI SERVICE=LIST .....	G-38
Example .....	G-40
\$RMDBAPI SERVICE=SET .....	G-41
Example .....	G-42
\$RMEVENT .....	G-43
Example .....	G-44
\$RMSTSET .....	G-45
Example .....	G-46
\$RECALL SERVICE=SET .....	G-47
Example .....	G-48
\$RECALL SERVICE=GET .....	G-49

Example.....	G-50
\$RECALL SERVICE=ACTION.....	G-51
Examples.....	G-52
\$REDBAPI SERVICE=CREATE.....	G-53
Example.....	G-55
EventView Definition Field Names.....	G-55
Ruleset Fields.....	G-56
Message Rule Fields.....	G-57
\$REDBAPI SERVICE=DELETE.....	G-61
Example.....	G-62
\$REDBAPI SERVICE=GET.....	G-63
Example.....	G-64
\$REDBAPI SERVICE=LIST.....	G-65
Example.....	G-66

## **Appendix H SSI DD SUBSYS Support..... H-1**

About DD SUBSYS.....	H-2
Using DD SUBSYS.....	H-2
Implementing DD SUBSYS.....	H-2
Who Can Use DD SUBSYS Facilities?.....	H-4
NMSSI DD SUBSYS Support.....	H-4
Enabling DD SUBSYS Support in NMSSI.....	H-5
Stopping the SSI.....	H-5
General NMSSI DD SUBSYS Syntax.....	H-6
Supported Function Names.....	H-6
Common Functions.....	H-7
COPY Facility.....	H-7
FILTER Facility.....	H-9
Carriage Control.....	H-10
WTO Facility.....	H-10
WTO DD SUBSYS Operands.....	H-11
PPISEND Facility.....	H-12
PPISEND DD SUBSYS Operands.....	H-12
PPIRECV Facility.....	H-14
PPIRECV DD SUBSYS Operands.....	H-14
USER, USERI, and USERO Facilities.....	H-16
USERx DD SUBSYS Operands.....	H-17

FILTER Exit API .....	H-18
APF Authorization .....	H-19
Abnormal Termination (ABEND) .....	H-19
Dynamic Allocation .....	H-20
Calling Details .....	H-20
Sample Filter Exit .....	H-24
USERx Facility API .....	H-24
APF Authorization .....	H-25
Abnormal Termination (ABEND) .....	H-25
Supported I/O Requests .....	H-26
Dynamic Allocation .....	H-26
Calling Details .....	H-27
Sample USER Program Exit .....	H-33
Sample FILTER Exit—UTIL0037 .....	H-33
UTIL0037 Processing .....	H-34
Control File Format .....	H-34
Filtering Processing .....	H-34
Control Statements .....	H-35
Sample Filter Table .....	H-37
Sample USER Exit—UTIL0038 .....	H-37
UTIL0038 Processing .....	H-37
UTIL0038 USERI Processing .....	H-38
UTIL0038 USERO Processing .....	H-38
UTIL0038 Encryption .....	H-38

**Appendix I    Sysplex Support..... I-1**

Cloning Support .....	I-2
Registration of a Region With the Sysplex Automatic Restart Manager .....	I-2
Restart Status Messages .....	I-3
Enabling the Generation of the Restart Status Messages .....	I-3
Restart Status Message Syntax .....	I-4

**Glossary**

**Index**



---

# Figures and Tables

Figure 2-1.	Alert Monitor Administration Menu.....	2-4
Figure 2-2.	Alert Monitor : Interface Definition Panel.....	2-5
Figure 2-3.	Alert Monitor : Email a Trouble Ticket Panel.....	2-6
Figure 2-4.	Alert Monitor : Custom Trouble Ticket Panel.....	2-7
Figure 2-5.	Alert Monitor : Trouble Ticket Data Entry Definition Panel ...	2-8
Figure 2-6.	Result of the TT Command When No Problem Ticket Data Entry Definition Has Been Created.....	2-9
Figure 2-7.	Customized Trouble Ticket Details Panel .....	2-9
Figure 2-8.	Problem Ticket Data Entry Definition—Example.....	2-10
Figure 2-9.	Logon Scripts Panel.....	2-14
Figure 3-1.	Relationship between UAMS and NPF .....	3-5
Figure 3-2.	Relationship Between UAMS and SAF .....	3-7
Figure 3-3.	Levels of Access and Privilege—Example .....	3-14
Figure 3-4.	UAMS Update Report.....	3-15
Figure 3-5.	User Description Panel .....	3-16
Figure 3-6.	Resource Monitor Profile Panel.....	3-17
Figure 3-7.	Graphical Monitor Profile Panel.....	3-19
Figure 3-8.	Alert Monitor Profile Panel .....	3-20
Figure 3-9.	Console Consolidation Profile Panel .....	3-21
Figure 3-10.	VTAM Domain Consolidation Profile.....	3-23
Figure 3-11.	User Profile Panel Display List.....	3-24
Figure 5-1.	Multisystem Operation .....	5-3
Figure 5-2.	Multisystem Link Configurations .....	5-5
Figure 5-3.	Linking Regions .....	5-9
Figure 5-4.	Multi-System Support Menu.....	5-10
Figure 5-5.	Remote System Identification Panel.....	5-12

Figure 5-6.	Confirm Database Synchronization Panel .....	5-12
Figure 5-7.	Monitoring the Progress of Knowledge Base Synchronization .....	5-13
Figure 5-8.	Browsing the Task Log .....	5-14
Figure 5-9.	Linked Regions Panel .....	5-17
Figure 5-10.	Browsing a System Link Definition .....	5-18
Figure 5-11.	Confirming a Region Unlink Operation .....	5-19
Figure 5-12.	Transmitting Knowledge Base Records .....	5-21
Figure 5-13.	Redeveloping a System Image in a Stand-alone Region .....	5-24
Figure 5-14.	Updating the Active System Image .....	5-25
Figure 6-1.	Confirming a Log Swap.....	6-2
Figure 7-1.	The Logical State Attributes Table .....	7-3
Figure 7-2.	The Automated Mode Attributes Table .....	7-4
Figure 7-3.	The Manual Mode Attributes Table.....	7-5
Figure 7-4.	The SNA Resource Logical State Normalization Table .....	7-6
Figure 8-1.	Single-column Status Monitor Display.....	8-2
Figure 8-2.	Two-column Status Monitor Display.....	8-2
Figure 8-3.	List of Status Monitor Display Formats.....	8-3
Figure 8-4.	Status Monitor Display Format List Description.....	8-5
Figure 8-5.	Supplied 1-column Status Monitor Display Format .....	8-6
Figure 8-6.	Variable Name Aliases Used in the Supplied 1-column Status Monitor Display Format.....	8-8
Figure 8-7.	Multiscreen Status Monitor Display Format.....	8-9
Figure 8-8.	Alert Monitor Display Format List Description .....	8-11
Figure 9-1.	Adding a Value to a Prompt List .....	9-3
Figure 9-2.	Defining a Field Prompt.....	9-4
Figure 10-1.	A Macro List .....	10-3
Figure 10-2.	Updating a Macro Definition .....	10-4
Figure 11-1.	A Command List.....	11-3
Figure 11-2.	Browsing the Command Details Panel .....	11-4
Figure 11-3.	Browsing the Command Prompting, Confirmation and Validation Panel.....	11-5
Figure C-1.	Console Settings for a Production Region .....	C-5
Figure D-2.	Sample NMSSI Logo Screen.....	D-7
Figure D-3.	NMSSI Menu .....	D-8

Table 3-1.	NPF Members—Samples.....	3-6
Table 3-2.	Component Class Numbers.....	3-36
Table 4-1.	RMIWTO06 WTOR Message—Reply Parameters.....	4-3
Table A-1.	Security Settings for \$RMADMIN.....	A-2
Table A-2.	Security Settings for \$RMOPER.....	A-3
Table A-3.	Security Settings for \$RMNOPER.....	A-5
Table A-4.	Security Settings for \$RMMON.....	A-6
Table A-5.	Security Settings for \$RMBUSER.....	A-7
Table B-1.	ICS Parameter Groups.....	B-2
Table D-1.	Non-VTAM Terminal Sessions Status.....	D-9
Table E-1.	SMF Record Header Format.....	E-2
Table E-2.	EventView Statistics SMF Record Format.....	E-3
Table E-3.	ResourceView and ServiceView SMF Record Format.....	E-4
Table G-1.	System Image Field Names Used in the \$RMDBAPI Procedure.....	G-22
Table G-2.	Resource Field Names Used in the \$RMDBAPI Procedure.....	G-23
Table G-3.	EventView Ruleset Field Names Used in the \$REDBAPI Procedure.....	G-56
Table G-4.	EventView Message Rule Field Names Used in the \$REDBAPI Procedure.....	G-57.



# Part I

---

## **Administering Automation Services**



---

# Introduction

The key issue in operations is to ensure service availability for end users. As networks and systems become increasingly diverse and complex, the challenge for an enterprise is to ensure the availability of critical services and resources when required. The products that use Automation Services help you operate, and maintain the availability of, large, centralized systems, or systems distributed over networks.

**Note**

Some products (for example, NetMaster for SNA and NetSpy) uses only a subset of these services. See the product guides.

**This chapter contains the following topics:**

- What Automation Services Offers
- Features
- About this Guide
- What You Should Already Know
- Notational Conventions

See the *Automation Services Common User Guide* for detailed information about other functions.

---

## What Automation Services Offers

Automation Services supports a suite of operations automation and management application products that run on IBM OS/390 mainframe computers. A subset of these products also run on VM/ESA, Hitachi VOS3, and Fujitsu MSP mainframe computers. The products can be implemented on one or more computers in an organization, depending on the automation requirements of the organization.

Automation Services enables you to automate the operation of the information system (IS) resources in your organization. It establishes a *region* that uses the information stored in a knowledge base to automate the operations processes in your organization.

The knowledge base contains the operations policies and methods for the delivery of services to your organization. System images, which define the resources that are to be controlled by the region, are part of the knowledge base. Each system has its own system image. (A system is the equivalent of a single operating environment, such as an OS/390 production system.) A system image definition contains all the resources required to automate the operation of that system.

By using the knowledge base, the region maintains the availability of the resources that deliver services to your organization. The operators are relieved of routine operations tasks, freeing them to concentrate on abnormal operations conditions.

For example, your organization might need to make a production system available to users at 0800 hours each weekday, bring the system down at 2300 hours to start backups, then make the production system available again at 0800 hours the next business day. The region can automate this and many other operations activities.

The region also minimizes the impact of a failure on the users of a service. If a failure occurs while the service is idle, automation can recover the service before a user requires that service. That is, the users do not know about the failure and see the service as continuously available.

---

## Features

Automation Services offers many features that help you implement and administer your region. These features are described in the following sections.

### AutoAssist

AutoAssist is a computer-assisted initialization facility that enables you to implement a region rapidly and easily. You are also able to easily customize both region and automation parameters at a later stage.

When you first log on to a region, you need to set various parameters to get the product up and running. The Initialization and Customization Services (ICS) helps you set up these parameters. An initial dialogue is supplied for the first time user, to walk you through the customization process. No coding is required, but you are prompted to supply required parameter values—such as dataset names—and given the opportunity to supply optional parameter values (depending on which setup option you select).

### User Management

Automation Services supplies a user profile facility that enables you to set up user profiles with associated User Access Maintenance Subsystem (UAMS) user ID definitions and to attach the user to an appropriate UAMS group.

Automation Services provides two levels of security:

- The first level of security uses UAMS, or a user-defined partial or full security exit. This provides user ID and password validation, and some basic security. Security exits are provided for the RACF and CA-ACF2 security packages on OS/390 systems.
- The second level of security secures the functions and resources of Automation Services. This is provided by the Network Partitioning Facility (NPF), or the IBM System Authorization Facility (SAF), used by external security packages.

## Multisystem Support

Automation Services provides focal point management to support multisystem operation (that is, management at a focal point with subordinates feeding information to it) as follows:

- Peer-to-peer architecture—supports full connectivity between multiple regions. Automation Services regions linked in this way are known as focal point regions. (A stand-alone region is also regarded as a focal point region.)

When regions are communicating with each other, authorized users can monitor and control all managed resources from any terminal connected to a focal point region.

- Subordinate—enables you to reduce the amount of traffic in your multisystem environment. You link subordinates to focal point regions that provide central monitoring and control. A subordinate has visibility and control of the locally managed resources only.
- Independent operation—each region can run independently of other regions. If no communication links are available, each region still provides full monitoring, control, and automation of its own managed resources. If one region fails, automation continues for resources that are managed by the other regions.

You can also transmit various types of data from a stand-alone region to another region.

- Communication access methods—multisystem links can support the following communication access methods: EndPoint Services (EPS), TCP/IP, and VTAM.
- Multisystem knowledge base—Automation Services automatically maintains synchronization between linked knowledge bases, with automatic recovery in the event of link failure.

### Note

For linked focal point regions, synchronization is complete and the focal point knowledge bases duplicate each other.

For linked subordinates, synchronization is complete only to the extent of the relevant definitions in the knowledge base. For example, a subordinate knowledge base does not contain all system images. A subordinate knowledge base contains only those images that represent the environment the subordinate is managing.

## Product Administration

Automation Services provides the following implementation and administration functions:

- The ability to customize display attributes—display attribute tables list the logical states of services and resources and the display attributes used by the *status* and *graphical monitors* to represent these logical states. (The status monitor displays resource status information line by line; the graphical monitor displays resource status information graphically in the form of colored *traffic light* icons.)

There are three display attribute tables:

- Logical State: lists the display attributes used by the status and graphical monitors to show statuses.
  - Automated Mode: lists the logical states assigned to services and resources when their operations are automated.
  - Manual Mode: lists the logical states assigned to services and resources when their operations are not automated.
- The ability to update prompt lists. These are lists of valid values (from which you choose one value) that are displayed when you enter a question mark (?) in a prompted field. Prompted fields are identified by the presence of a plus sign (+) at the start of the input field.
  - The ability to write your own NCL procedures or macros for use in Automation Services processes.
  - The ability to define commonly used monitor commands—these commands are also NCL procedures.

## System Programming

There are a couple of facilities available to system programmers:

- Automation Services application programming interface (API)
- Subsystem Interface (SSI) DD SUBSYS support (MSP and OS/390)

## Automation Services Application Program Interface

The API provides the following procedures:

- \$RMCALL
- \$RMDBAPI
- \$RMEVENT
- \$RMSTSET
- \$RECALL
- \$REDBAPI

These procedures enable sources external to Automation Services to call Automation Services functions and retrieve Automation Services data.

## The DD SUBSYS Advanced Facility

DD SUBSYS is a facility that allows an authorized subsystem to provide new access methods to existing programs. In effect, the subsystem appears to these programs as if it is a set of files, and in this way is able to provide data to, or receive data from, the programs.

For example, assume that you have a third-party program that writes an event log while it is running. The information on this event log would be very useful for system automation. Normally, however, the data written to this log cannot be accessed by another program until the program producing the log is shut down. DD SUBSYS allows you to intercept all data written to the file, while logging is taking place. You can then use the WTO facility (supported by Automation Services) to send the data to operator terminals.

## Management Services

Management Services provides a central core of basic functions and services to various products, including those products supported by Automation Services.

The Management Services component supports operations and includes the following services:

- Activity logs, which record the activities that occur within a region. You can search through a log by time, date, or text strings.
- Messages and codes database, which provides information about error messages and codes (for example, VSAM OPEN macro return codes).
- Network Control Language (NCL), which enables you to develop procedures for performing specific tasks. Automation Services commands and macros are written in NCL.

---

## About this Guide

This guide describes how to implement, customize, and manage a region that uses Automation Services.

Automation Services enables data center staff to automate the management of services and resources within the organization, and manage the message flow on all systems.

This guide is written for data center staff who perform the following tasks:

- Customize the Automation Services software
- Control access to the Automation Services software
- Start and stop a region
- Customize and maintain the user environment

---

## What You Should Already Know

Users include administrators, programmers, and operations analysts. You should have an understanding of the operations environment at your site before reading this guide.

Refer to this guide during product implementation to assist with the customization of the product.

---

## Notational Conventions

The following sections describe some of the notations used in this guide.

### Variables Representing Data Set Names

This guide uses the following variable representation in data sets:

*dsnpref*

Represents the data set prefix used during product installation.

*rname*

Represents the name of the region.

*ccvvv*

Represents the product component identifier used during installation (for example, AS500).

## Product Specific Information

This guide supports the mainframe products that use Automation Services. Where it is necessary to differentiate product specific information, the product is identified in parentheses as follows:

<b>For ...</b>	<b>The identifier is ...</b>
Unicenter NetMaster File Transfer Management (NetMaster for File Transfer)	FT.
Unicenter NetMaster Network Automation (NetMaster Automation)	NMA.
Unicenter NetMaster Network Management for SNA (NetMaster for SNA)	SNA.
Unicenter NetMaster Network Management for TCP/IP (NetMaster for TCP/IP)	IP.
Unicenter NetSpy Network Performance (NetSpy)	NS.
Unicenter SOLVE:Operations Automation (MSP)	MSP.
Unicenter SOLVE:Operations Automation (SOLVE:Operations for OS/390)	OS/390.
Unicenter SOLVE:Operations Automation (VOS3)	VOS3.
Unicenter SOLVE:Operations Automation for CICS (SOLVE:Operations for CICS)	CICS.

# 2

---

## Initializing, Implementing, and Customizing a Region

This chapter provides information about how to implement common Automation Services features in a region. For product-specific implementation information, see the product *Implementation Guide*.

**This chapter contains the following topics:**

- Initialization and Customization Services
- Initializing a Region for the First Time
- Implementing a SOLVE:Problem Automatic Problem Recording Environment
- Implementing the Alert Monitor Problem Ticket Interface
- Dealing With Initialization Failures
- Setting Up Checkpoint Restart of a System Image
- Customizing External Applications Access
- Generating an INI Procedure

---

## Initialization and Customization Services

Initialization and Customization Services (ICS) provides the facility for the review and update of parameter groups. No coding is required.

### What are Parameter Groups?

System parameters are grouped by category (such as *Security*) in logical parameter groups, to simplify the process of initializing and customizing a region. The parameter groups are listed in Appendix B, *Parameter Groups*.

Groups of individual parameters translate into one or more of the following:

- SYSPARMS that determine how your region functions. (If you want to learn about SYSPARMS, see the *Management Services Administrator Guide*.)
- Global variables that are used by various NCL applications to control their functions.
- Local parameters that define how to implement actions associated with parameter groups (for example, specifying data sets to be allocated).

You can apply the following actions to listed parameter groups:

- **S** (Select) or **B** (Browse) to browse parameter group details (if you have the authority, you can switch from Browse to Update mode).
- **H** (Help) to obtain help on a parameter group.
- **L** (ILog) to view the initialization and customization log for a particular parameter group.

You can also view the ICS log by using the ILOG command.

- **SD** (Set Default) to reset the parameter group values to the original default values. To apply the default values, you need to update the parameter group.
- **U** (Update) to update parameter group details (you can action changes made to parameter groups by pressing F6, but these changes may not be saved when you log off—see help).

---

## Initializing a Region for the First Time

At the end of the installation and setup tasks, you should have defined an initial user in the region.

To perform first time customization, follow the procedure in the product *Implementation Guide*. The following sections include some additional features that you might want to implement in your region.

---

## Implementing a SOLVE:Problem Automatic Problem Recording Environment

The region can add problem records to a SOLVE:Problem application.

If you want to enable automatic recording of specified problems in SOLVE:Problem, perform the following tasks:

- Task 1—Copying the \$RMPB07S NCL Procedure
- Task 2—Defining Users in the SOLVE:Problem Region
- Task 3—Defining Links

### Task 1—Copying the \$RMPB07S NCL Procedure

Copy the \$RMPB07S NCL procedure from the *dsnpref.ASvvv.ASTEXEC* data set to the NCL procedures library (normally TESTEXEC) in the region in which the SOLVE:Problem application is running. (*dsnpref* is the data set prefix used during the installation of this product.)

**Note**

On VM systems, copy the procedure from the SLVMAINT 293 G-disk to the 292 F-disk.

### Task 2—Defining Users in the SOLVE:Problem Region

Define the following user IDs to the SOLVE:Problem region:

- Automation Services BSYS background user ID *xxxxBSYS*, where *xxxx* is the domain ID of the product region
- IDs of the users who may raise problem tickets manually from the alert monitor in the product region

## Task 3—Defining Links

In the SOLVE:Problem region, define a link to each product region from which you want to receive problem tickets as follows:

```
DEFLINK TYPE=APPC LUNAME=acb-name LINK=link-name
```

*acb-name* is the ACB name of the product region, and *link-name* is a name that identifies the link.

---

## Implementing the Alert Monitor Problem Ticket Interface

The alert monitor provides an interface that enables you to raise problem tickets for alerts.

The alert monitor supports the following interfaces for raising problem tickets:

### Note

NetMaster Network Automation on VM systems, and SOLVE:Operations Automation on MSP and VOS3 systems do not support the e-mail interface.

- Electronic mail, where an e-mail describing the problem can be sent to a problem management application or to a particular person. This method can be used to send tickets to multiple problem management applications.
- Custom, where you can write your own NCL code to deliver the ticket to an application by whatever means you choose.

To implement this feature, define the problem ticket interface between your region and the alert monitor (see the section, *Defining a Problem Ticket Interface*, on page 2-5).

If you want the operator to supply information when requesting the creation of a ticket, you also need to set up the problem ticket data entry definition (see the section, *Setting up the Problem Ticket Data Definition*, on page 2-8).

To perform these tasks, enter **/ALADMIN** to display the alert monitor Administration Menu (see Figure 2-1).

*Figure 2-1. Alert Monitor Administration Menu*

```
I - Define Trouble Ticket Interface
D - Define Trouble Ticket Data Entry
H - Set Alert History Logging Parameters
F - Define Filters
L - Define List Formats
X - Exit
```

## Defining a Problem Ticket Interface

To define a problem ticket interface between your region and the alert monitor, complete the following steps:

- Step 1. On the Alert Monitor Administration Menu, select option **I** - Define Trouble Ticket Interface to display the Alert Monitor : Interface Definition panel (see Figure 2-2).

Figure 2-2. Alert Monitor : Interface Definition Panel

```
SOLVPROD----- Alert Monitor : Interface Definition -----  
Command ===>  
  
Press F6 to confirm the Interface type  
Interface Type .....+ _____  
-----
```

- Step 2. In the Interface Type field, specify the type of interface that you want to define. To obtain a selection list of valid values, enter ? in this field.
- Step 3. Press F6 (Action). A panel is displayed where you can define your interface. The type of panel displayed varies, depending on the interface type that you specified. For further details, see the section, *Defining an E-mail Problem Ticket Interface*, on page 2-5 and the section, *Defining a Custom Problem Ticket Interface*, on page 2-7.

## Defining an E-mail Problem Ticket Interface

**Note** NetMaster Network Automation on VM systems, and SOLVE:Operations Automation on MSP and VOS3 systems do not support the e-mail interface.

If you specified **EMAIL** as the interface type on the Alert Monitor : Interface Definition panel, pressing F6 (Action) displays the Alert Monitor : Email a Trouble Ticket panel.

To define your e-mail interface, complete the following steps:

- Step 1. Enter values in the input fields in the top section of the panel.  
You can use F1 (Help) to obtain information about completing these fields.
- Step 2. Complete the Enter Mail Text Below section of the panel. This is free-form.  
You can use F1 (Help) to obtain information about completing this section.

Figure 2-3 shows an example of a completed panel. The `&$USRNAME` variable, together with a data entry definition, enables you to prompt users to enter the address of the mail receiver. (For information about how to create the data entry definition, see the section, *Setting up the Problem Ticket Data Definition*, on page 2-8.)

**Note** Even if you have not used an `&$USR` prefixed variable, it is recommended that you create a data entry definition (see the section, *Considerations*, on page 2-9).

Figure 2-3. Alert Monitor : Email a Trouble Ticket Panel

```
SOLVPROD----- Alert Monitor : Email A Trouble Ticket ----Columns 001 074
Command ==>                                     Function=Update Scroll ==> PAGE

Mail Address          &$USRNAME_____
Host Name      (IBM)  ALMVS1__
SMTP Node Name (IBM)  ALMVS1__
SMTP Job Name  (IBM)  SMTP32__
SMTP DEST Id (TCPaccess) _____
Exit Procedure Name  _____
Subject            &$AMDESC_____

Enter Mail Text Below

**** ***** TOP OF DATA *****
0001 Alert Creation Date      : &$AMDATE
0002 Alert Creation Time     : &$AMTIME
0003
0004 Severity                 : &$AMSEVERITY
0005
**** ***** BOTTOM OF DATA *****

F1=Help      F2=Split      F3=File      F4=Save      F5=Find      F6=Change
F7=Backward  F8=Forward      F9=Swap      F10=Left     F11=Right    F12=Cancel
```

- Step 3. Press F3 (File) to save your interface definition and return to the alert monitor Administration Menu.

## Defining a Custom Problem Ticket Interface

If you specified **CUSTOM** as the interface type on the Alert Monitor : Interface Definition panel, pressing F6 (Action) displays the Alert Monitor : Custom Trouble Ticket panel.

To define your custom interface, complete the following steps:

- Step 1. In the Procedure Name input field, enter the name of your NCL procedure for delivering tickets.
- Step 2. In the Enter Parameters Below section of the panel, specify any parameters that you want the NCL procedure to receive. This section is free-form.

You can use F1 (Help) to obtain information about completing this section.

Figure 2-4 shows an example of an interface that uses the distributed SOLVE:Problem exit, \$RMPB06S, to send tickets to a SOLVE:Problem region with the ACB name SOLVPROB. (For more information about the SOLVE:Problem interface, see the section, *Implementing a SOLVE:Problem Automatic Problem Recording Environment*, on page 2-3.)

*Figure 2-4. Alert Monitor : Custom Trouble Ticket Panel*

```
SOLVPROD----- Alert Monitor : Custom Trouble Ticket ----Columns 001 074
Command ==>                                     Function=Update Scroll ==> CSR

Procedure Name  $RMPB06S

                Enter Parameters Below

**** ***** TOP OF DATA *****
0001 ACBNAME=solvprob
**** ***** BOTTOM OF DATA *****
```

- Step 3. Press F3 (File) to save your interface definition and return to the alert monitor Administration Menu.

## Setting up the Problem Ticket Data Definition

If you want the operator to supply information when requesting the creation of a problem ticket, you need to set up the ticket data entry definition as follows:

- Step 1. On the alert monitor Administration Menu, select option **D** - Trouble Ticket Data Entry to display the Alert Monitor : Trouble Ticket Data Entry Definition panel.
- Step 2. In the free-format data entry section of the panel, enter the data entry definition for the panel that the operator will use when creating a ticket.

You can use F1 (Help) to obtain information about completing this section.

Figure 2-5 shows an example of a definition that prompts the operator to identify the receiver of the ticket.

*Figure 2-5. Alert Monitor : Trouble Ticket Data Entry Definition Panel*

```
SOLVPROD----- Alert Monitor : Trouble Ticket Data Entry Definition -----
Command ==>                                     Function=Update Scroll ==> PAGE

**** ***** TOP OF DATA *****
0001 FIELD NAME=$USRNAME
0002 VALUE="Problem@sydney.enterprise.com"
0003 DESC="Send Email to:"
0004 COMMENT="(name for e-mail)"
0005 REQUIRED=YES
0006 LENGTH=40
**** ***** BOTTOM OF DATA *****
```

- Step 3. Press F3 (File) to save your trouble ticket data entry definition and return to the alert monitor Administration Menu.



Figure 2-8 shows the data entry definition.

Figure 2-8. Problem Ticket Data Entry Definition—Example

```
SOLVPROD----- Alert Monitor : Trouble Ticket Data Entry Definition -----
Command ==>                                     Function=Update Scroll ==> CSR
AMTTDED08 TROUBLE TICKET DATA ENTRY DEFINITION SAVED
**** ***** TOP OF DATA *****
0001 FIELD NAME=$USRX
0002 VALUE=
0003 DESC="Press F6 to send the ticket"
0004 COMMENT=
0005 REQUIRED=NO
0006 LENGTH=0
**** ***** BOTTOM OF DATA *****
```

---

## Dealing With Initialization Failures

If you log on to a region that has not completed initialization, ICS displays the System Initialization In Progress dialog to indicate progress, and to assist you in identifying and rectifying any problems.

Fatal errors occur (for example, you are unable to log on) if either or both of the following are unavailable:

- Panel libraries
- MODS control files

## Using the System Initialization In Progress Dialog

The System Initialization In Progress dialog shows you the current initialization status and whether actions associated with parameter groups have failed. In the following text, the term *parameter group* refers to both the value assigned to a parameter and to the action or actions associated with that value (such as setting SYSPARMS, allocating data sets, and so on).

The System Initialization In Progress dialog enables you to:

- List only failed parameter groups
- List all outstanding parameter groups; that is, those that have failed, are queued to be applied, or are in the process of being applied
- List all parameter groups, including those that have run successfully
- View the system messages in the ICS log

The parameter group lists are dynamic action lists. You can apply the following actions to listed parameter groups:

- **S** or **B** (Browse) to browse parameter group details
- **H** (Help) to view Help for a parameter group
- **U** (Update) to update parameter group details
- **AC** (Action) to action a parameter group
- **L** (ILog) to view the associated initialization and customization log
- **I** (Ignore) to indicate to the system that it should ignore a failed parameter group, and proceed to run dependent parameter groups

See the online help for additional assistance.

### Note

An action can only be performed against an already completed parameter group or a failed parameter group.

### Note

Ignoring parameter groups is not recommended; consider carefully before applying this action.

When you correct an error by updating an incorrect parameter group record, you have to action that parameter group before processing can continue (unless you apply the Ignore action). Action the parameter group in one of the following ways:

- By pressing F6 (the Action key) when you finish updating the parameter group
- By applying the **AC** (Action) action to the listed parameter group

---

## Setting Up Checkpoint Restart of a System Image

Checkpoint restart is the capability of a region to remember overrides placed on the resources in the active system image by operators—this will survive a stop and restart of your region.

Checkpoint restart is set in the \$RM AUTOIDS parameter group. This parameter group also allows you specify the following:

- Warm Load—retrieves the previously stored manual overrides.
- Cold load—does not retrieve any previously stored manual overrides associated with the image.

You can warm load a system image only if checkpoint restart is enabled. When checkpoint restart is enabled, any manual overrides placed on the members of the loaded image are preserved and will be reapplied on the next load of that image.

With checkpoint restart enabled, warm load is the default. If you want to perform a cold load, you must respecify it each time you perform the load. To perform a cold load every time, disable checkpoint restart.

---

## Customizing External Applications Access

The external applications function allows you to access other products, such as NetMaster for SNA or SOLVE:Configuration, to assist in monitoring resources. It allows you to specify access to external applications for network management, configuration management, problem management, and help desk.

To enable access to your external applications from the region, perform the following steps:

- Step 1. Enter **/ICS** from the primary menu to display the Customization Parameters panel.
- Step 2. Enter **U** (Update) beside the **\$RM EXTAPPLS** parameter and press **ENTER** to display the **EXTAPPLS - External Application Access** panel.
- Step 3. Enter the full screen terminal and the LU 1 terminal prefixes (which are the terminal name prefixes for the VTAM APPLS as defined during your installation).
- Step 4. Enter values in the Application ID field, and optionally in the Logon Script and Logon Data fields for each application that you want to access. Use **F8** (Forward) to access all the application definition panels.
- Step 5. Press the **F4** (Save) key to save the changes and the **F3** (File) key to return to the Customization Parameters panel.

Once the applications are specified, they are accessed by using the following predefined commands:

<b>Command</b>	<b>Application</b>
CFG	Configuration Management
HD	Help Desk
PRB	Problem Management
XNM	External Network Management

## Navigating to an External Application

To customize where you log on to your external application, you can use a logon script. A logon script allows you to automate the logon process. There are logon scripts supplied for the following products:

<b>Application</b>	<b>Logon Script</b>
NetMaster for SNA	RMSCPTNM
SOLVE:Problem	RMSCPTPM
SOLVE:Configuration	RMSCPTCM
SOLVE:Problem Help Desk	RMSCPTHM

It is possible to customize these logon scripts to meet your own application requirements.

To customize a logon script, complete the following steps:

- Step 1. Enter the **/ASADMIN.S** path to display the Logon Scripts panel (shown in Figure 2-9).

*Figure 2-9. Logon Scripts Panel*

```
Select Option ===>

  C - Generate Configuration Management Script
  H - Generate HelpDesk Script
  N - Generate Network Management Script
  P - Generate Problem Management Script
  BC - Browse Configuration Management Script
  BH - Browse HelpDesk Script
  BN - Browse Network Management Script
  BP - Browse Problem Management Script
  X - Exit
```

- Step 2. Enter the letter of the required Generate option at the Select Option prompt to display the Logon Recording : Recording Details panel.
- Step 3. Specify the output string. This is the string that appears on the primary menu of the application that you are logging on to.
- Step 4. Specify the resource name string. This string will be recognized during recording as the point to substitute a resource name for which the command is to be executed.
- Step 5. Specify the stop key. The default is F15. This key ends the logon recording session.
- Step 6. Press the F6 (Action) key to begin the logon recording.

Step 7. Log on to the application, and do one of the following:

- Proceed to the required location, and press the stop key.
- Proceed to a location where a resource name is required, and enter the resource name string as specified in Step 4. Continue further if desired, and then press the stop key.

This displays the Logon Recording : Save Script panel.

Step 8. Enter the data set details and press the F6 (Generate) key to generate the script.

The generated script is displayed for your review.

Step 9. Press the F6 (Confirm) key to write the script to the data set. You are then returned to the application session.

---

## Generating an INI Procedure

### Note

The VM operating system does not support the INI procedure generation function. However, it *can* use an existing INI procedure.

Parameter group records store information about parameters that have been changed by customization. Customized parameter group records are stored in the VFS data set. If you want to store the information in the form of a partitioned or a sequential data set, you need to generate an INI procedure.

When you generate an INI procedure, the customized parameter group records are converted into procedural (NCL) code, which is exported to the nominated partitioned data set.

To generate an INI procedure, complete these steps:

- Step 1. Enter **/INIT.G** from the primary menu to display the Generate INI Procedure panel.
- Step 2. Identify the required output data set member.
- Step 3. Press F6 (Action) to generate an NCL procedure. The generated NCL procedure is displayed for your review.
- Step 4. When you are happy with the displayed procedure, press F6 (Confirm) to save the procedure to the nominated output data set.

### Caution

You can edit an INI procedure by using an external package such as TSO/ISPF, but we recommend that *only the customized values* contained in this procedure be updated. The INI procedure comprises standard NCL logic. Further NCL logic can be added. However, this is not required or recommended.

If updating, always create a backup PDS member.

## Using an INI Procedure to Propagate Customized Parameter Group Records to Other Regions

You can use a generated INI procedure to propagate customized parameter group records to other regions; for example, from a test to a production region.

To propagate customized parameter group records to other regions:

- Ensure that the generated INI procedure is saved in a PDS that is allocated in the COMMANDS DD of the BSYS background user ID.
- Specify the name of the member where the procedure is saved, as the value of the INIFILE parameter in your RUNSYSIN member.

### Note

If a valid INI procedure is specified in the INIFILE parameter when region initialization occurs, the customization values contained in the nominated procedure override the customization values stored in the VFS data set. The VFS data set is also permanently updated to reflect the values specified in the INI procedure.

### Note

If you use an initialization INI procedure to start up your region and you changed the settings of a parameter group, you must regenerate the procedure if you want the changes applied at subsequent startups of the region.



---

# Implementing Security

This chapter describes how to implement security.

**This chapter contains the following topics:**

- About Security
- Generating Background User IDs
- Customizing Parameters That Affect Security
- User Profiles—Levels of Access and Privilege
- Defining Users by Using User Profiles
- Maintaining User Profiles
- Defining a User ID With Special Security Requirements
- Authorizing VM Programmable Operator Interface Users
- Controlling Access to Functions and Resources by Using NPF
- Implementing Security Through an External Security Package
- Controlling Access to Functions and Resources by Using an External Security Package
- Securing Data Set Members

---

## About Security

Security is implemented on two levels:

- Signon access
- Access to functions and resources

### Controlling Signon Access

Signon access to a region is controlled by one of the following:

- The User ID Access Maintenance Subsystem (UAMS)
- An external security package that performs some or all of the security functions through a full or partial security exit.

## UAMS

By using UAMS, you can define user IDs for each user of Automation Services, providing logon and password checking. User IDs can be added, deleted, or updated.

You can either define each user's user ID separately or add users with the same security requirements by using a UAMS group.

Automation Services supplies sample group definitions that are generated during installation. These groups and their characteristics are described below:

- \$RMADMIN—administrator—this group of users has access to all administrative functions, such as adding user IDs and user profiles. An administrator has access to all menu options and is authorized to delete database records.
- \$RMOPER—operator—this group of users has access to a restricted subset of functions. An operator does not have access to all menu options and is not authorized to delete database records.
- \$RMNOPER—network operator—this group of users has similar access as an operator. Network operators can manage network operations but are not authorized to manage system operations. Use this group for network operators managing SNA resources from a NetMaster for SNA or a NetMaster Network Automation region.
- \$RMMON—monitor—this group of users has access to a restricted subset of menu options, and can browse, but not update or delete database records. These users can display information about monitored resources but cannot act on those resources.
- \$RMBUSER—Background User—this group of users has region or engine component authorization. *Do not* modify the supplied \$RMBUSER group definition, because this could impede the operation of the automation engine.

See the sections, *Defining Users by Using User Profiles*, on page 3-15, and *Defining a User ID With Special Security Requirements*, on page 3-25 for more information.

## External Security Packages

External security packages, such as RACF, CA-ACF2, or CA-Top Secret can provide minimal or full security checking.

If your organization has an external security package, access to that package is provided through one of the following types of exit:

- *Partial security exit*—password and logon access maintenance is controlled by the external security package while UAMS stores the user definitions.
- *Full security exit*—all security functions are maintained and stored by your external security package.

The following sample security exits are provided with Automation Services and can be found in the SMP target zone library, *dsnpref.INSTAL*:

**Note**

On VM systems, these exits are on the SLVMAINT 193 C-disk.

<b>Sample Exit</b>	<b>Description</b>
CCRACFFX	Full security exit for RACF
CCACF2FX	Full security exit for CA-ACF2
NMSAFPX	SAF partial security exit

The NMSAFPX partial security exit is a SAF based security exit that supports UTOKENS. SAF is the IBM System Authorization Facility and is the agreed standard for the encoding of requests that require security checking.

**Note**

SAF is documented in the IBM *External Security Interface (RACROUTE) Macro Reference* manual. See the documentation for your security package to see whether the package supports SAF-formatted calls.

For OS/390 systems, it is recommended that you use the NMSAFPX partial security exit as it supports RACF, CA-ACF2, and Top Secret security packages.

The product supplies a partial security exit. To use the exit, specify the SEC=PARTSAF parameter in the RUNSYSIN member.

If you write your own exit, link it to create the load module. Place the module in the load library, or another library concatenated to the load library through the STEPLIB DD statement in the started task JCL. Change the JCL parameters to include the SEC=*load-module-name* parameter. This causes the security exit to be used.

See Appendix A, *Security Settings*, for information about the minimum security settings for different types of users. See the *Management Services Administrator Guide* for information about the security exit support provided by a region.

## Controlling Access to Functions and Resources

Access to the functions and resources is controlled using the Network Partitioning Facility (NPF), your external security package, or both.

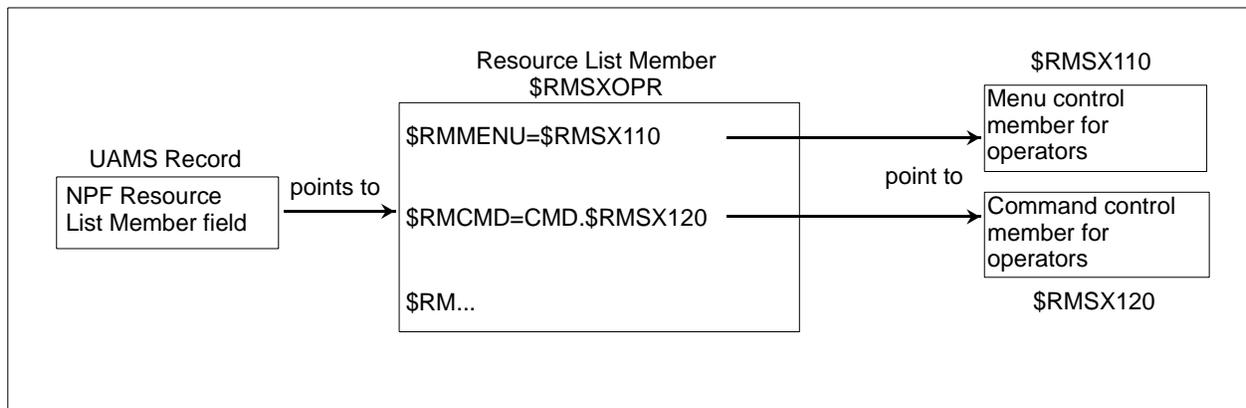
### Network Partitioning Facility (NPF)

The facility contains the access permissions for user's requests to menus, resources, or commands. This is enabled when an NPF resource list member is specified in a UAMS ID definition.

By specifying structured strings, access can be restricted or allowed to menu options, databases, system images, commands, and initialization and customization parameter groups. These strings are stored in resource tables and referenced by the NPF resource list members.

Figure 3-1 shows the relationship between UAMS and NPF.

Figure 3-1. Relationship between UAMS and NPF



Automation Services provides sample NPF members that contain predefined tables that permit or restrict access for different groups of user. These NPF members and their content are shown in Table 3-1. For more information about these members, see the comments within the members.

**Caution**

The NPF members must have names that start with \$RMSX. If you rename a member, ensure that its name has the correct prefix.

*Table 3-1. NPF Members—Samples*

<b>NPF Member</b>	<b>Content</b>
\$RMSXADM	Permissions for administrators
\$RMSXOPR	Permissions for operators
\$RMSXNOP	Permissions for network operators
\$RMSXMON	Permissions for monitors
The following members (with names prefixed by \$RMSXS) requires an external security package:	
\$RMSXSAF	Special indicator—NPF is to be bypassed in favor of an external security package
\$RMSXSAD	Permissions for administrators—combined NPF and external security package
\$RMSXSOP	Permissions for operators—combined NPF and external security package
\$RMSXSNO	Permissions for network operators—combined NPF and external security package
\$RMSXSNO	Permissions for network operators—combined NPF and external security package
\$RMSXSMO	Permissions for monitors—combined NPF and external security package

When a user accesses a menu or tries to issue a command, NPF is called. The region checks the NPF member specified in the user ID definition and its corresponding permissions. The region then responds by showing particular menu options, or allowing or disallowing the command.

By using NPF, you can also restrict certain users to certain groups of resources. For example, one operator can influence all the resources in REGION1 only while another operator can influence all the resources in REGION2. Any attempt by the first operator to influence the resources in REGION2 will be rejected. The same restriction can apply to the messages from these regions.

**Note**

The region does not perform read access controls; that is, all users are able to browse all data. Users who attempt to update data they are not authorized to update are presented with a warning message, and the data is not modified.

See the section, *Controlling Access to Functions and Resources by Using NPF*, on page 3-26 for more information.

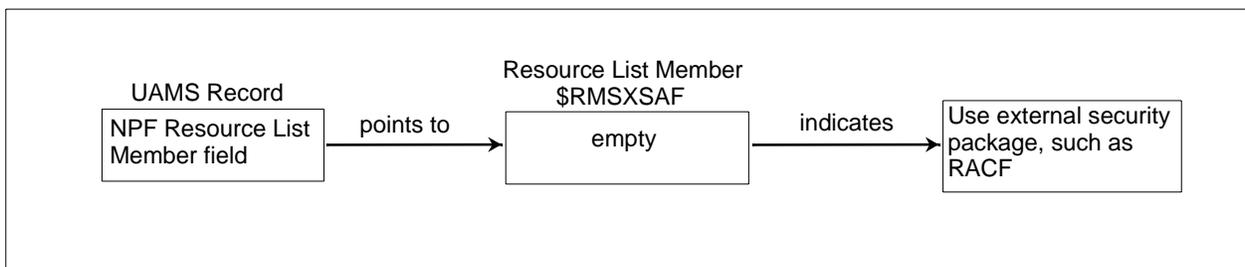
### External Security Options

Your external security package can provide Automation Services security if it supports SAF. It can provide security on its own or in conjunction with NPF.

SAF contains the access permissions for user's requests to menus, resources, and commands. This is enabled when \$RMSXS-prefixed member is specified in the NPF Resource List Member field of a UAMS ID definition.

Figure 3-2 shows the relationship between UAMS and SAF within an external security package. The \$RMSXS-prefixed member can either be empty (as shown in the figure) or contain NPF statements. Sample members are listed in Table 3-1 on page 3-6.

Figure 3-2. Relationship Between UAMS and SAF



Sample security profiles are provided for each of the supported external security packages in the *dsnpref.ASvvv.ASSAMP* dataset, as shown below. *dsnpref* is the dataset prefix and *vvv* is the product version used during the installation of the product. You can copy, customize, and integrate the appropriate profile into your security package.

**Note**

On VM systems, these profiles are on the SLVMAINT 293 G-disk.

<b>Security Profile</b>	<b>Content</b>
\$RMSXACF	Permissions for all groups of users for CA-ACF2 versions earlier than Version 6
\$RMSXAC6	Permissions for all groups of users for CA-ACF2 Version 6 and later
\$RMSXRCF	Permissions for all groups of users for RACF
\$RMSXTSS	Permissions for all groups of users for CA-Top Secret versions earlier than Version 5
\$RMSXTS5	Permissions for all groups of users for CA-Top Secret Version 5 and later

Your external security package is called when a user accesses a menu or tries to issue a command. The region then passes the associated structured string to your external security package as a SAF formatted call. The security package responds by showing specified menu options, or allowing or disallowing the command.

See the section, *Controlling Access to Functions and Resources by Using an External Security Package*, on page 3-34 for more information.

---

## Generating Background User IDs

The \$RMBUSER group definition for background regions is defined when a region starts the first time. The following UAMS background user definitions (where *nnnn* is the domain ID) are also generated and linked to the UAMS group:

User ID	Description
<i>nnnn</i> AOMP	AOM procedure
<i>nnnn</i> BLOG	Logger
<i>nnnn</i> BMON	Monitor
<i>nnnn</i> BSVR	Server
<i>nnnn</i> BSYS	System
<i>nnnn</i> LOGP	Log procedure

If any of the default background group definitions are not defined in UAMS, you can create them by running \$NMUAINI from the message monitor. The procedure to do this is:

- Step 1. Enter **CMD** from the primary menu to display the Command Entry panel.
- Step 2. Enter **\$NMUAINI** at the command prompt (===>).

## Background User Considerations for Existing UAMS File

If you set up your region by using a pre-existing UAMS file in which the background users are already defined for your region, those background user definitions are not replaced. To enable the new region to work correctly, you must update those background user definitions by associating the definitions to the \$RMBUSER group ID. You can do this by completing the following steps:

- Step 1. Enter `=/UAMS` to access the UAMS maintenance function.
- Step 2. Update each of the background user IDs by entering \$RMBUSER in the Group ID field.
- Step 3. Press F3 to file the changes, and again to exit.
- Step 4. Enter **CMD** to display the Command Entry panel, and enter the **SUBMIT *background-user-id* SIGNON** command for each of the background regions to invoke the changes.

The \$RMBUSER group ID provides the minimum security settings needed for a background user. However, additional settings can be added to meet your requirements.

**Note**

This process can be done at any stage, including during initial customization when the region is first started.

---

## Customizing Parameters That Affect Security

Review the following Initialization and Customization Services (ICS) parameter groups for security:

ICS parameter	Description
CMDREPLS	Command replacement
SEC SHIPPING	Ship UAMS maintenance

### Command Replacement

The CMDREPLS parameter group specifies which commands are to be intercepted and have an NCL procedure of the same name started instead of the command being executed. This allows you to perform additional security checking on these commands.

#### Caution

If you change the default set of replacements, the functioning of some of the supplied applications may be impacted.

For more information about command replacement, see the section, *Controlling Access to Management Services Commands from the Message Monitor*, on page 3-32.

### Synchronizing Updates Across Linked Regions

To automatically update UAMS records across all active linked regions, you need to enable the automatic propagation facility.

#### Note

If automatic propagation is enabled, it occurs when a user ID is added or updated by using the method described in the section, *Defining Users by Using User Profiles*, on page 3-15.

Synchronization depends on whether you make an update from a focal point region or a subordinate region as follows:

#### If the update is in a ... The update is synchronized across ...

Focal point region      All active linked regions that are enabled for this feature.

Subordinate region      Only those active linked focal point regions that are enabled for this feature.

Enabling or disabling the update of UAMS records across multiple regions is the function of parameter group \$RM SECSHIPPING. This parameter group can be set or altered subsequently by:

- Step 1. Entering **/ICS** from the primary menu to display the Customization Parameters panel.
- Step 2. Applying the **U** (Update) action to \$RM SECSHIPPING, which is located under the SECURITY category.

With the SECSHIPPING - Ship UAMS Maintenance panel displayed, the following settings can be made:

- Respond **YES** to both questions.  
This allows all add, update, delete, *and* password change operations for UAMS records to be propagated to linked regions. (This setting is for regions that do not share a UAMS file and do not use the RACF distributed password update facility.)
- Respond **YES** to the question Ship to Linked Systems? and **NO** to the question Including Password Changes?  
This allows all add, update and delete operations for UAMS records to be propagated to the linked regions. Update requests from linked regions are processed, but changes to the password field are not. (This setting is for regions that do not share a UAMS file but use the RACF distributed password update facility.)
- Respond **NO** to both questions.  
This means that no UAMS records changes are propagated. If an update is requested from a remote region via this facility, it is refused. Note that, if linked regions share a UAMS file, you should choose this setting. Otherwise, you will get error messages when updating shared values on the User Description panel of the user profile.

UAMS updates are sent to linked regions immediately, for security reasons. A UAMS update report is displayed immediately, indicating the success or failure of those updates.

## Troubleshooting

Possible reasons for a remote region update not working include the following:

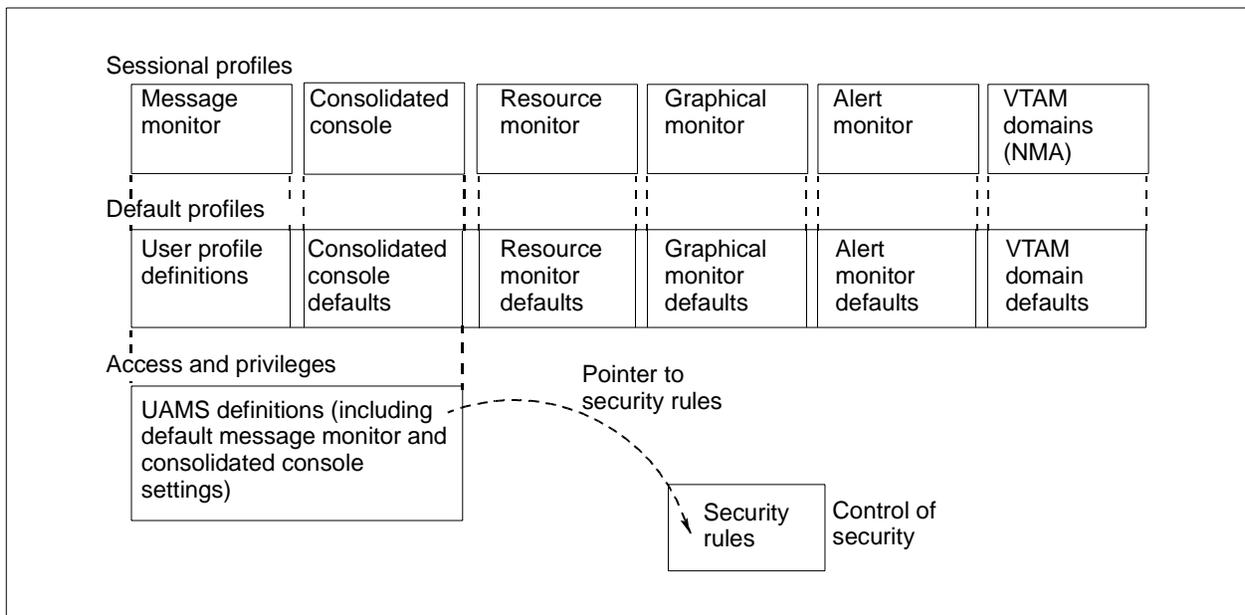
- The region is not profiled for remote updates. To profile it for remote updates, specify **YES** in the Ship to Linked Systems? field of its SEC SHIPPING parameter group.
- The link or remote region is not active. (Because UAMS update records are not written to a staging file, no record of the update is retained, and the UAMS record in the remote region is not updated.)
- The user does not have UAMS administration authority on the remote region.
- The record or database is locked.
- The UAMS record does not exist. This condition occurs when an administrator updates a user profile record, without providing a new initial password, and there is no associated UAMS record for the remote region. The administrator can remedy this by supplying an initial password. This results in the automatic generation of a UAMS record for the remote region, and resets the users password across all regions.
- You based the user access on a customized user group that does not exist in the remote region.

## User Profiles—Levels of Access and Privilege

Figure 3-3 shows examples of user access and privilege. Profiles that are specific to a particular product are identified in parentheses.

- At the lowest level, UAMS uses the security rules to control basic functions, security, and logon access to the region. Only a system administrator should have the authority to maintain UAMS definitions.
- At the next level, user profiles contain default settings (which are within any UAMS limitations) for the operational environment. Users can update their own profiles, and the system administrator can update any user profile.
- At the highest level, the specific environments seen by a user are determined by the defaults set in that user's profile record. A user can, however, change those defaults (within any UAMS or user profile limitations) for the duration of a work session.

Figure 3-3. Levels of Access and Privilege—Example



---

## Defining Users by Using User Profiles

User profile information controls what menus, resources, and messages they see on monitors and consoles. User profile records are automatically propagated to all linked regions when created.

To assist with ease of administration, a UAMS user ID definition is automatically generated when a user profile is created for a user and a group ID is specified.

To create a profile for a new user, complete the following steps:

- Step 1. Enter **/ASADMIN** from the primary menu to display the Automation Services Administration Menu.
- Step 2. Select option **UP** - User Profiles to display the User Profile List.
- Step 3. Press F4 (Add) to add a new user profile. The action presents you with the first panel in the user profile definition. The following sections describe some of the panels. Use F8 (Forward) to scroll to each new panel.
- Step 4. File or save the new record.

A UAMS Update Report (see Figure 3-4) is displayed if the following apply:

- You have completed fields on the User Description panel.
- The \$RM SECSHIPPING parameter group is set in the current region to enable automatic propagation of UAMS updates. See the section, *Customizing Parameters That Affect Security*, on page 3-11 for more information on \$RM SECSHIPPING.

*Figure 3-4. UAMS Update Report*

```
SOLVPROD----- Automation Services : UAMS Update Report -----
Command ==>                               Function=Browse Scroll ==> PAGE

          A UAMS update request was sent to all linked
          regions for user BROWNP with the following results:

Region    Message
-----
SOLV13    RM350008 UAMS ADD PROCESSED SUCCESSFULLY
SOLV14    RM350403 UAMS ADD PROCESSED SUCCESSFULLY
SOLV15    RM350403 UAMS ADD PROCESSED SUCCESSFULLY

***** BOTTOM OF DATA *****
```

### Note

Once you have defined one user profile, you can use the **C** (Copy) action to duplicate an existing user profile and change the values for another user in the copied record as required.

## Specifying a User's Details

You can specify a user's details by using the User Description panel. These details are the same as those in the user's UAMS user ID definition. An example of this panel is shown in Figure 3-5.

Figure 3-5. User Description Panel

```
SOLVPROD----- Automation Services : User Description -----
Command ==> file                                         Function=ADD

. User Description -----
|
| User ID ..... BROWNP__
| Initial Password ...
|
| User Name ..... Peter Brown_____
| User Location ..... Operations_____
| Telephone Number ... ext 222_____
|
| Group ID .....+ $RMOPER_
| Group Name ..... Operator Group
|-----
```

### Note

If the user is already defined to UAMS, you can enter an equals sign (=) in the User Name, User Location, Telephone Number, and Group ID fields, and the values for these fields are derived from the UAMS record. Also, you do not need to complete the Initial Password field.

If, however, you enter new values in any of these fields, the user's UAMS record is updated with the new values.

## Customizing a User's Resource Monitor Display

### Note

Besides the resource monitor profile, there are profiles for other types of monitors that you may customize. Depending on the products running in your region, you have available different types of monitors.

You can customize a user's resource monitor display by using the Resource Monitor Profile panel. An example is shown in Figure 3-6.

Figure 3-6. Resource Monitor Profile Panel

```
SOLVPROD----- Automation Services : Resource Monitor Profile -----BROWNP
Command ==>                                     Function=UPDATE

. Resource Monitor Display Control -----
Resource Filter .....+ JOB_____
Monitor List Format for ... 1 Column ...+ DEFAULT_ 2 Columns ...+ DEFAULT_
                          3 Columns ..+ DEFAULT_ 4 Columns ...+ DEFAULT_
                          5 Columns ..+ DEFAULT_
Monitor Columns ..... 1_ (1-5)
Resource Sort Criteria ...+ _____
                          + _____
                          + _____
Extended Display ..... ON_

F1=Help      F2=Split      F3=File      F4=Save
F7=Backward  F8=Forward      F9=Swap
F11=Panels   F12=Cancel
```

The panel displays the following information:

- Name of the resource monitor filter
- Display format

The definitions of the fields on this panel are as follows:

**Resource Filter**

Identifies the filter used by the resource monitor.

This is the filter used when the user accesses the monitor. Once in the monitor, the user can change the filter (see the *Automation Services Common User Guide*).

**Monitor List Format for ...**

Specifies the format to use for each of the column displays. Enter ? to display a list of valid formats and select one.

The product supplies a number of predefined formats. You can define your own formats from the List Definition List panel. To access the list, enter the /ASADMIN.L path. See Chapter 8, *Customizing Display Formats*, for information about how to define status monitor formats.

**Monitor Columns**

Specifies the number of resources to display across your screen. The number must be in the range 1 through 5. The default is 1.

**Resource Sort Criteria**

Specifies the criteria used to sort the displayed resources. You can specify up to three criteria. Resources are first sorted by the first criterion specified, then by the second (if a second criterion is specified), and so on.

**Extended Display**

Controls the display of user-defined extended displays. The valid values are ON and OFF. The default is ON.

# Customizing a User's Graphical Monitor

You can specify the icon panel to be displayed when a user first accesses the graphical monitor by using the Graphical Monitor Profile panel. An example is shown in Figure 3-7.

Figure 3-7. Graphical Monitor Profile Panel

```
SOLVPROD----- Automation Services : Graphical Monitor Profile -----BROWNP
Command ==>                                     Function=UPDATE

.- Graphical Resource Monitor Primary Panel -----
| Panel Name .....+ $RMDEFAULT__
|-----

F1=Help      F2=Split    F3=File      F4=Save
F7=Backward  F8=Forward   F9=Swap      F11=Panels   F12=Cancel
```

## Customizing a User's Alert Monitor

You can customize a user's default alert monitor display by using the Alert Monitor Profile panel. An example is shown in Figure 3-6.

Figure 3-8. Alert Monitor Profile Panel

```
SOLVPROD----- Automation Services : Alert Monitor Profile -----BROWNP
Command ==>                                                    Function=UPDATE

. Alert Monitor Display Control -----
|
| Monitor List Filter .....+ _____
|
| Monitor List Format .....+ _____
|
|-----
```

The panel enables you to specify defaults for the following:

- Name of the alert monitor filter that restricts the displayed alerts (You can define the filters at the Filter Definition List panel. To access the list, enter the **/ALADMIN.F** path.)
- Name of the display format that determines what alert information is displayed (You can define the format at the List Definition List panel. To access the list, enter the **/ALADMIN.L** path.)

## Customizing a User's Message Monitor Profile

You can customize a user's message monitor (or Operator Console Services) profile by entering the relevant information on the following four panels:

- Message Monitor Screen Control
- Message Monitor Command Control
- Message Monitor Message Receipt
- Message Monitor Message Formatting

### Note

Customized values (that is, values other than the default) are set for the message monitor only if the \$RMCCOCS procedure is run. The procedure is specified in the user ID definition.

See the online help for assistance in entering appropriate values on these panels.

## Entering a Command String

The Message Monitor Command Control panel includes an Initial Command field. You can use this field to enter a command string that is issued when the user selects the message monitor.

## Customizing a User's Consolidated Console

You can specify the message profiles to activate when a user accesses the consolidated console by using the Console Consolidation Profile panel. Figure 3-9 is an example of the panel.

Figure 3-9. Console Consolidation Profile Panel

```
SOLVPROD----- Automation Services : Console Consolidation Profile -----
Command ==>                                     Function=UPDATE

-----
| This panel allows you to specify which of the available message profiles |
| you want automatically activated when you view your consolidated console |
| window. Only the profiles actually selected will be activated, but you   |
| can select as many profiles as you like, and in any combination.         |
|-----|
|-----|
| Profile Name  ID      User      Global      Description |
| ALLHOMSGS    42     DISABLED INACTIVE    HO all messages definition |
| SYSTEM$SOLV1 69     DISABLED ACTIVE     All messages from this system |
|-----|
```

The panel displays the list of message profiles available to the user. (These message profiles are specified as the console routing codes in the user definition and correspond to the message profile IDs).

For each profile, the panel displays the following:

- Name of the profile
- ID of the profile
- User status indicating whether the profile is enabled to the consolidated console

**Note**

Even if a profile is enabled, it is not effective unless its global status is also ACTIVE.

- Global status indicating whether the profile is defined to be active (that is, whether the profile can be loaded)
- Description of the profile

A profile has a status of either **DISABLED** or **ENABLED**. To change the status, complete these steps:

- Step 1. Press **F10** (ScrLst) to change the list to a scroll list, so that you can use **F7** (Backward) and **F8** (Forward) to scroll through the list (instead of scrolling through the user profile panels).
- Step 2. Type **D** next to the profiles you want to disable and **E** next to the profiles you want to enable, and press **ENTER**.
- Step 3. Press **F3** (OK) to accept the settings; press **F3** (File) to file the settings.

## Customizing a User's SNA Network Summary Display

If you are using the NetMaster Network Automation product, you can specify the VTAM domains to be monitored in the SNA network summary display by using the VTAM Domain Consolidation Profile panel. Figure 3-10 is an example of the panel.

Figure 3-10. VTAM Domain Consolidation Profile

```
SOLVPROD--- Automation Services : VTAM Domain Consolidation Profile -----USER01
Command ===>                                     Function=UPDATE

-----
This panel allows you to specify which of the available Domains are to be
automatically combined in the SNA Resource Summary View.
-----

Net ID      Domain Name
SDINET1    SYDVTM01

-----
F1=Help      F2=Split    F3=File     F4=Save
F7=Backward  F9=Swap     F10=LstRegn F11=Panels  F12=Cancel
```

The panel displays the list of domains that are included in the SNA network summary display.

To change which domains are included and excluded, complete the following steps:

- Step 1. Press F10 (LstRgn) to list the domains that are managed by the region.
- Step 2. Type **S** or **I** next to the domains to be included and **X** next to the domains to be excluded.
- Step 3. Press F3 (OK) to accept the settings; press F3 (File) to file the settings.

---

## Maintaining User Profiles

User profile records are maintained by applying actions to items on the User Profile list. You can update, copy, or delete listed profile records.

## Updating User Profiles

To update a user profile definition:

- Step 1. Enter **/ASADMIN.UP** from the primary menu to display the User Profile List.
- Step 2. Apply the **U** (Update) action to the item you want to update.

The Panel Display List panel, an example of which is shown in Figure 3-11, is displayed.

*Figure 3-11. User Profile Panel Display List*

```
SOLVPROD----- Automation Services : Panel Display List -----
Command ==>                                     Scroll ==> PAGE

                                     Use 'S' to select panel/s to be displayed

Panel Description
User Description
Resource Monitor Profile
Graphical Monitor Profile
Alert Monitor Profile
Message Monitor Screen Control
Message Monitor Command Control
Message Monitor Message Receipt
Message Monitor Message Formatting
Console Consolidation Profile
VTAM Domain Consolidation Profile
**END**
```

- Step 3. Select the panel you want to update.
- Step 4. Update the fields on this panel as required. If you want to update further fields on other user profile panels, use F7 (Backward) and F8 (Forward) to move between panels.
- Step 5. File (F3) the updated definition.

### Note

You can customize parts of your own user profile from certain associated panels. For example, you can customize your resource monitor profile from the status monitor by using the **PROFILE** command.

Users who have sufficient authority can update their own user profiles. Enter **U.UP** on the primary menu to display the list of user profile panels, then select the panel to be updated. (The sequence of panels can be scrolled by using F7 and F8.)

## Deleting a User Profile Definition

If you want to delete a user profile record *and* its associated UAMS record, apply the **D** (delete) action to that item on the User Profile list. To delete the user profile record while retaining the UAMS record, apply the **DP** (delete profile) action.

---

## Defining a User ID With Special Security Requirements

Normally, there is no need to create user IDs, as an associated user ID is generated automatically when a user profile is added for a user and a group ID is specified. Individual user IDs should only be defined when security requirements not covered in the four group IDs are needed.

To define an individual user ID for a new user, complete the following steps:

- Step 1. Enter the **/UAMS** shortcut to access the UAMS primary menu.
- Step 2. Select option **A** from the UAMS : Primary Menu to display the first of several panels of user ID definition details prescribing a user's access to various features.

### Note

If the user ID you are defining is similar to another ID, you can save time by copying an existing ID to a new user you wish to add. Do this by selecting option **C** (Copy) from the primary menu, then specify the user ID for the existing user as the model. UAMS retrieves the details for that user ID and then enters Add mode. You must now enter the new user ID name and change any fields required, before filing the new definition under the new name.

- Step 3. Enter the type of user ID you are defining in the Definition Type field on the panel.  
The default is **USER**, meaning that an individual user ID is being defined, otherwise enter **GROUP** in this field if you wish to define a group user ID.
- Step 4. Enter the required information on each following panel, scrolling forward (F8) to review the next panel.
- Step 5. File (F3) the new user ID definition once all panels have been reviewed and the required attributes specified. You are returned to the UAMS : Primary Menu.

### Note

If you define user IDs using the above method, automatic propagation to all linked regions will not occur. Automatic propagation only occurs when defining user profiles.

## Defining Users For An External Security Package

If you are using an external security package with a full security exit, you need to define a user ID for each user. The minimum security settings for each of the supplied group IDs are provided in Appendix A, *Security Settings*. You can use this information to help you define individual user IDs or other group definitions.

---

## Authorizing VM Programmable Operator Interface Users

If you want to authorize an interface user to issue system commands from the region, authorize the user on the UAMS : AOM VM Details panel.

---

## Controlling Access to Functions and Resources by Using NPF

As discussed earlier, you can control user access to functions and resources by specifying NPF resource list members. These are: \$RMSXADM, \$RMSXOPR, \$RMSXNOP, and \$RMSXMON.

To enable a user's security to be controlled by one of these members, you need to specify the member in the NPF Resource List Member field on the OCS details panel of the user ID definition.

### Note

If you have an external security package, see the sections, *Implementing Security Through an External Security Package*, on page 3-34 and, *Controlling Access to Functions and Resources by Using an External Security Package*, on page 3-34.

You can alter sample members to meet you own security requirements by changing the structured strings that are stored in the NPF resource tables. Following is a list of the NPF resource tables and the functions that can be secured by each:

<b>NPF Resource Table</b>	<b>Function Secured</b>
\$RMMENU	Menu options
\$RMDB	Databases
\$RMSYS	System images
\$RMCMD	Commands
\$RMICS	Initialization and customization parameter groups

The rest of this section explains how to alter the NPF resource tables.

### Caution

Use the method documented here to control access to resources and functions. Do *not* use the command authority function, because changing the authority level of commands can interfere with the operation of the region.

## Controlling Access to Menu Options

Access to menus and their options is controlled by using the \$RMMENU table.

To allow access to all menu options, specify the following:

```
$RMMENU=*.*
```

To restrict access to menus and options, specify the following:

```
$RMMENU=$RMSXnnn
```

where \$RMSXnnn is the control member for menu options for one user group. In this control member, you need to list all the menus and options using the following format is used:

<i>RM.menu-id.option-code</i>
-------------------------------

### *menu-id*

Identifies the menu. The ID of the main Primary Menu is \$NM001. To display the ID of another menu, enter **MENUID** at the ==> prompt.

### *option-code*

Identifies the option (for example, A for the Administration and Definition option on the main Primary Menu).

To indicate that certain menu options are invalid, you need to comment them by placing an asterisk (\*) beside them. To make the menu option valid again, uncomment the option by removing the asterisk.

## Controlling Access to the Knowledge Base

Access to the knowledge base is controlled by the \$RMDB table. Controlling access to the knowledge base allows you to control the type of access a user has to definitions by systems, classes, and resources.

## Controlling the Type of Access

The type of access is controlled by specifying the actions that can be performed on systems, classes, and resources. If no restriction is required, specify the following:

```
$RMDB=ACT . *
```

To restrict the type of access allowed, specify the following:

```
$RMDB=ACT . $RMSXnnn
```

where *\$RMSXnnn* is the control member for the type of access to databases for one user group. In this control member, you need to list the actions that are available, and comment or uncomment them as required. The valid actions are CREATE, DELETE, and SET.

## Controlling Access by System Images

If all systems are to have the type of access defined above, specify the following:

```
$RMDB=SYS . *
```

To restrict the defined access to only certain systems, specify the following:

```
$RMDB=SYS . $RMSXnnn
```

where *\$RMSXnnn* is the control member for systems with restricted access for one user group. In this control member, you need to list the systems that will have the defined access.

## Controlling Access by Classes

If all classes are to have the type of access defined above, specify the following:

```
$RMDB=CLS . *
```

To restrict the defined access to only certain classes, specify the following:

```
$RMDB=CLS . $RMSXnnn
```

where *\$RMSXnnn* is the control member for classes with restricted access for one user group. In this control member, you need to list the available classes, their short names, and their description, and comment or uncomment them as required.

## Controlling Access by Resources

If all resources are to have the type of access defined above, specify the following:

```
$RMDB=RSC . *
```

To restrict the defined access to only certain resources, specify the following:

```
$RMDB=RSC . $RMSXnnn
```

where *\$RMSXnnn* is the control member for resources with restricted access for one user group. In this control member, you need to list the resources that will have the defined access.

## Controlling Access to System Images

Access to system images is controlled by the \$RMSYS table.

To allow access to all system images, specify the following:

```
$RMSYS=SYS . *
```

To restrict access to certain system images, specify the following:

```
$RMSYS=SYS . $RMSXnnn
```

where *\$RMSXnnn* is the control member for access to system images for one user group. In this control member, you need to list the system images to which access is allowed.

## Controlling Access to Commands

Access to commands is controlled by the \$RMCMD table. Access to the following groups of commands can be controlled:

- Automation Services commands
- System commands
- Management Services commands

It is also possible to restrict the commands that can be performed against systems and resources.

## Controlling access to Automation Services Commands

To allow access to all Automation Services commands, specify the following:

```
$RMCMD=CMD . *
```

To restrict access to particular Automation Services commands, specify the following:

```
$RMCMD=CMD . $RMSXnnn
```

where *\$RMSXnnn* is the control member for access to Automation Services commands for one user group. In this control member, you need to list the commands, their classes, and their descriptions, and comment or uncomment them as required.

## Controlling Access to System Commands

To allow access to all system commands, specify the following:

```
$RMCMD=SYSCMD . *
```

To restrict access to particular system commands, specify the following:

```
$RMCMD=SYSCMD . $RMSXnnn
```

where *\$RMSXnnn* is the control member for access to system commands for one user group. In this control member you need to list the commands, their classes, and their descriptions, and comment or uncomment them as required.

## Controlling Access to Management Services (MS) Commands

To allow access to all MS commands, specify the following:

```
$RMCMD=NMCMDB . *
```

To restrict access to particular MS commands, specify the following:

```
$RMCMD=NMCMDB . $RMSXnnn
```

where *\$RMSXnnn* is the control member for access to MS commands for one group of user. In this control member, you need to list the commands, their classes, and their descriptions, and comment or uncomment them as required.

## Controlling Access to Commands Issued Against Systems

If the commands defined above are to be issued against all systems, specify the following:

```
$RMCMDSYS.*
```

To restrict the defined commands to only certain systems, specify the following:

```
$RMCMDSYS.$RMSXnnn
```

where *\$RMSXnnn* is the control member that controls the systems against which defined commands can be issued for one user group. In this control member, you need to list those systems against which the defined commands can be issued.

## Controlling Access to Commands Issued Against Resources

If the commands defined above are to be issued against all resources, specify the following:

```
$RMCMDRSC.*
```

To restrict the defined commands to only certain resources, specify the following:

```
$RMCMDRSC.$RMSXnnn
```

where *\$RMSXnnn* is the control member that controls the resources against which defined commands can be issued for one user group. In this control member, you need to list those resources against which the defined commands can be issued.

## Controlling Access to Management Services Commands from the Message Monitor

When you issue a MS command from the message monitor, you issue the command under the control of your command authority level and the external security profile of the region. You do *not* normally issue the command under the control of the NPF member specified in your UAMS record.

If you want to issue a MS command from the message monitor, but under the control of the specified NPF member, replace the command with an NCL procedure.

Automation Services provides the following NCL procedures to replace MS commands: ALLOCATE, EDIT, FSTOP, OPSYS, ROUTE, SHUTDOWN, SUBMIT, SYSCMD, and SYSPARMS.

If you need to create other replacement NCL procedures, do the following:

Step 1. Create an NCL procedure in the security PDS with the same name as the command you want to replace. (See the section, *Securing Data Set Members*, on page 3-42 for information about the security PDS.)

Step 2. Ensure the NCL procedure contains the following:

```
-EXEC $RMSXTPL cmdname &ALLPARMS
&IF &RETCODE EQ 0 &THEN +
  -cmdname &ALLPARMS
```

Step 3. Enter **/ICS** from the primary menu to display the Customization Parameters panel.

Step 4. Add your replacement NCL procedure name to the parameter group ID CMDREPLS in category SECURITY.

CMDREPLS can contain up to 21 entries. If you have more than 21 entries, place the command SYSPARMS CMDREPL=*cmdname* for each extra entry in the NMINIT procedure.

Step 5. Press F6 (Action) if you want to use the replacement NCL procedure immediately (otherwise, it will only be available after the region has been restarted.)

### Caution

The NPF security rule, \$RMCMD.REPLUNLD, controls whether a user can use the SYSPARMS UNLOAD command to unload the command replacement NCL procedures. Ensure that you provide sufficient security for the resource \$RMCMD.REPLUNLD, to prevent unauthorized unloading (disabling) of the listed NCL procedures.

## Controlling Access to Initialization and Customization Parameter Groups

Access to all initialization and customization parameter groups is controlled by the \$RMICS table.

To allow all types of access to initialization and customization parameter groups, specify the following:

```
$RMICS=*.*
```

To restrict the type of access to initialization and customization parameter groups, specify the following:

```
$RMICS=$RMSXnnn
```

where \$RMSXnnn is the control member that controls the type of access to initialization parameter groups for one user group. In this control member, you need to list the type of access, and the initialization and customization groups, and comment or uncomment them as required. The following format is used:

<i>action.parameter-group-name</i>
------------------------------------

### *action*

Specifies the type of access to the initialization and customization parameter groups. The access can be one or more of the following:

<i>action</i>	<b>Function</b>
GET	Gets parameter group.
SET	Files parameter group.
SETPARM	Actions parameter group.
UPDPARM	Updates parameter group.
BROPARM	Browses parameter group.

### *parameter-group-name*

Names the initialization and customization parameter group.

## Changing an NPF Table

If you make changes to an NPF table, these changes are only activated when you have done the following:

- Executed the NPTAB *resource-group* REP=*table-name* command; for example, NPTAB \$RMMENU REP=\$RMSX110.
- Executed a SIGNON command, or logged off and logged back on again.

---

## Implementing Security Through an External Security Package

### Note

To implement security for functions and resources using an external security package, your system must support SAF.

To implement security for functions and resources using an external security package, you must run the batch job associated with the appropriate security profiles. The security profiles supplied are:

- \$RMSXACF and \$RMSXAC6 for CA-ACF2
- \$RMSXRCF for RACF
- \$RMSXTSS and \$RMSXTS5 for CA-Top Secret

To define the appropriate security profiles to an external security package, complete these steps:

- Step 1. Copy the required security profile member to the security PDS (which is the first library in the COMMANDS concatenation of libraries), as documented in the section, *Securing Data Set Members*, on page 3-42.
- Step 2. If necessary, modify the members to suit your requirements (see the section, *Controlling Access to Functions and Resources by Using an External Security Package*, on page 3-34 for information on how to do this).
- Step 3. Add a valid job card to run the batch job.
- Step 4. When the job has completed successfully, enter a \$RMSXS-prefixed member in the NPF Resource List Member field of your user's group or user ID definition, to indicate that an external security package is required to control security.

The security requirements for the sample groups—\$RMADMIN, \$RMOPER, \$RMNOPER, and \$RMMON—are now defined to your external security package and will apply to users attached to these groups.

---

## Controlling Access to Functions and Resources by Using an External Security Package

The supplied security members (discussed in the previous section) use SAF to control access to functions and resources. These security members can be modified to suit your own security requirements. Use the syntax specified in the following sections to specify your own requirements for access to menu options, the knowledge base, system images, Automation Services commands, system commands, and MS commands.

## Controlling Access to Menu Options

Specify the following to control access to menu options:

```
$RMMENU.menu-id.option-code
```

### *menu-id*

Identifies the menu. The ID of the main Primary Menu is \$NM001. To display the ID of another menu, enter **MENUID** at the ==> prompt.

### *option-code*

Identifies the option (for example, A for the Administration and Definition option on the main Primary Menu).

- Use an asterisk (\*) to represent all menu options; for example, \$RMMENU.*menu-id.\** means all options from *menu-id*.
- Use \$RMMENU.*\*\** to allocate access to all menus and their options.

### **Note**

The asterisk (\*) represents null, or one or more characters. Two asterisks (\*\*) represent any suffix. This may not apply to your security system, in which case, you need to use the equivalent wildcard character that does apply.

## Controlling Access to the Knowledge Base

The knowledge base contains definitions of Automation Services components. Use the following to control access to knowledge base definitions:

```
$RMDB.system-image-name.system-image-version.class-number.  
definition-name.action-type
```

### *system-image-name*

Names the system image.

### *system-image-version*

Identifies the version of the system image.

**class-number**

Identifies the class of component. Table 3-2 lists the valid classes. Classes that are specific to a particular product are identified in parentheses.

*Table 3-2. Component Class Numbers*

<b>class-number</b>	<b>Component Definition</b>
01	System image
02 (MSP, VOS3, CICS, FT, and OS/390)	Started task
03 (NMA)	SNA group
04 (CICS)	CICS transaction
05 (CICS)	CICS file
06 (CICS)	CICS database
07 (CICS)	CICS link
08 (FT)	CONNECT:Direct manager
09 (FT)	CONNECT:Direct monitor
10 (MSP, VOS3, and OS/390)	Initiator
11 (MSP, VOS3, and OS/390)	Printer
13 (MSP, VOS3, and OS/390)	Spool
14 (MSP, VOS3, and OS/390)	JES line
16 (MSP, VOS3, FT, and OS/390)	Direct access storage device (DASD)
17 (MSP, VOS3, CICS, OS/390, and NMA)	USRCLS—user-defined resource class
18 (MSP, VOS3, FT, and OS/390)	Tape or cartridge unit
19 (MSP, VOS3, CICS and OS/390)	Batch job
20 (MSP, VOS3, and OS/390)	Job entry subsystem (JES)
21	Internal resource
22 (FT)	FTS manager
23 (FT)	FTS monitor
24 (FT)	File transfer schedule
25 (FT)	FTP manager
26 (FT)	FTP monitor
27 (FT)	CONNECT:Mailbox manager

*(Sheet 1 of 3)*

Table 3-2. Component Class Numbers

<b>class-number</b>	<b>Component Definition</b>
28 (FT)	CONNECT:Mailbox monitor
29 (OS/390)	Sysplex component
30	Console message profile
35	User profile
36	Status monitor filter, resource group filter, SNA resource filter (NMA), file transfer ruleset (FT), or file transfer rule (FT)
38 (FT)	FTP policy ruleset or rule
40	Command
41	Logical state table
43 (IP and NS)	IP application
44 (IP and NS)	Channel Interface Processor
45 (IP and NS)	Router
46 (IP and NS)	Open Systems Adapter
47 (IP and NS)	Enterprise Extender
48 (IP and NS)	Communications storage manager
50	Availability map
51 (IP and NS)	TCP/IP stack
52 (IP and NS)	TCP/IP stack monitor
53 (IP and NS)	IP application server
54 (SNA)	Network control point monitor
55 (NMA)	SNA resource model template
57 (NMA)	VTAM state
60	Resource group
61	Service
62	CA-XCOM manager
63	CA-XCOM monitor

(Sheet 2 of 3)

Table 3-2. Component Class Numbers

<b>class-number</b>	<b>Component Definition</b>
70	Macro
71	Process
74 (FT)	Calendar criteria
75 (FT)	Calendar
76 (FT)	Calendar keyword
78 (FT)	Activity schedule
80	Icon
81	Icon panel
90	Prompt list
93	Ruleset
94	Message rule
95	Message group rule
96	Learnt message
98	Timer rule
9A	Included ruleset
9B	Initial action

(Sheet 3 of 3)

**Note**

Resource classes 30, 35 to 37, 40, 41, 60, 61, 70, 74 to 76, 78, 80, 81, 90, 93 to 96, 98, 9A, and 9B are independent of system images.

**definition-name**

Identifies the component definition.

**action-type**

Identifies the action to be performed on the definition. Valid values are:

<b>Action</b>	<b>Description</b>
CREATE	Create a new definition
DELETE	Delete an existing definition
SET	Update a definition

- Use an asterisk (\*) to represent all knowledge base components. The following example shows the ability to create started task resource definitions for all versions of *system-image-name*:

```
$RMDB.system-image-name.*.02.*.CREATE
```

- Use \$RMDB.\*\* to allow all knowledge base functions on all knowledge base components.

## Controlling Access to System Images

You can control which functions a user can perform on a system image. Use the following to control access to system image resources:

```
$RMSYS.system-image-name.system-image-version
```

### *system-image-name*

Names the system image.

### *system-image-version*

Identifies the version of the system image.

- Use an asterisk (\*) to represent all of a resource type; for example, \$RMSYS.\*.0001 indicates all version 1 system images.
- Use \$RMSYS.\*\* to allocate permission to perform all functions on all system images.

## Controlling Access to Automation Services Commands

You can control whether a user can issue an Automation Services command on an Automation Services component. Use the following to control the use of these commands:

```
$RMCMD.system-image-name.system-image-version.class-number.  
component-name.so-command-name
```

### *system-image-name*

Names the system image.

***system-image-version***

Identifies the version of the system image.

***class-number***

Identifies the class of component. See Table 3-2 on page 3-36 for a list of the valid class numbers.

**Note**

If you want to restrict the use of a command on SNA resources, specify 55 for the class number.

***component-name***

Identifies the component.

***so-command-name***

Names the command. See the section, *Accessing Command Definitions*, on page 11-3 for information on how to view the list of registered commands. Non-registered commands are:

<b>Command</b>	<b>Function</b>
DBSYNC	Synchronizes databases.
REPLUNLD	Enables a user to unload command replacement NCL procedures.
TRANSMIT	Transmits a system image or other database components.
SETUP	The express setup facility calls \$NMSEC with the command name SETUP to check if the user ID has authority to run the facility.

- Use an asterisk (\*) to represent all of a resource type. The following example shows the ability to issue commands relating to printers in all versions of *system-image-name*:

```
$RMCMD.system-image-name.*.11.*.*
```

- Use \$RMCMD.\*\* to allocate permission to perform all commands on all components in all system images.

## Controlling Access to System Commands

Use the following to control the use of system commands:

```
$RMSYCMD.system-command-name.operand-1.operand-2...operand-n
```

### *system-command-name*

Names the system command.

### *operand-n*

Specifies the operands of the command.

- Use `$RMSYCMD.**` to allocate permission to perform all system commands.

For example, to secure the MODIFY system command for the following MS commands in all regions:

- For the FSTOP command, use `$RMSYCMD.F*.FSTOP`.
- For the SHUTDOWN command, use `$RMSYCMD.F*.SHUTDOWN*`. The trailing asterisk ensures that every variation of the command is covered.

## Controlling Access to Management Services Commands

Use the following to control the use of MS commands:

```
$RMNMCMD.solve-command-name
```

### *solve-command-name*

Names the command. See the *Command Reference* for the list of MS commands.

- Use `$RMNMCMD.**` to allocate permission to perform all MS commands.

When you issue an MS command from the message monitor, you issue the command under the control of your command authority level and the external security profile of the region. You do not normally issue the command under the control of your own security profile.

If you want to issue an MS command from the message monitor, but under the control of your own security profile, replace the command with an NCL procedure. (See the section, *Controlling Access to Management Services Commands from the Message Monitor*, on page 3-32 for information on the NCL procedures and how to create new ones.)

### **Note**

This does not affect system commands that are already controlled by SAF.

## Controlling Access to Initialization and Customization Parameter Groups

Use the following to control the type of access to initialization and customization parameter groups:

```
$RMICS.action.parameter-group-name
```

### *action*

Specifies the type of access to the initialization and customization parameter groups. The access can be one or more of the following:

<i>action</i>	<b>Function</b>
GET	Gets parameter group.
SET	Files parameter group.
SETPARM	Actions parameter group.
UPDPARM	Updates parameter group.
BROPARM	Browses parameter group.

### *parameter-group-name*

Names the initialization and customization parameter group.

- Use the asterisk (\*) as a wild card. The following example shows the ability to retrieve all parameter groups:

```
$RMICS.GET.*
```

- Use `$RMICS.**` to allow all types of access to initialization and customization parameter groups.

---

## Securing Data Set Members

### **Note**

On VM systems, read:

- Minidisk for PDS
- SOLVE GCS for RUNSYSIN
- SLVMAINT 292 F-disk for TESTEXEC
- SLVMAINT 293 G-disk for *dsnpref.ASvvv.ASTEXEC*

The members that control a region need to be secured to ensure adequate security for the region. The library in which these members should be secured is called the security PDS. Only security personnel should be allowed access to the security PDS.

The security PDS is *not* created during the installation of your product, and *must* be created manually before you proceed to implement security. To establish a valid security PDS that secures all members controlling access to Automation Services functions, complete the following steps:

- Step 1. Create a security PDS, and ensure that it is the *first* library in the COMMANDS concatenation of libraries.

**Note**

The COMMANDS concatenation of libraries is in your RUNSYSIN member. The default first library is TESTEXEC.

- Step 2. Copy the following members from the *dsnpref.ASvvv.ASTEXEC* dataset into the security PDS, where *dsnpref* is the dataset prefix and *vvv* is the product version used during the installation of the product:

<b>Members</b>	<b>Description</b>
\$NMSEC	This member controls access to functions.
\$RMSXxxx	These members are sample SAF security profiles for RACF or CA-ACF2 (if you are using one of these packages to control access to the region) or NPF members (if you are using NPF to control access to the region).
ALLOCATE, EDIT, FSTOP, OPSYS, ROUTE, SHUTDOWN, SUBMIT, SYSCMD, and SYSPARMS	These members are command replacement NCL procedures; use them to control access by message monitor users to MS commands.

- Step 3. Copy any user-defined command replacement NCL procedures into the security PDS.
- Step 4. Restrict access to this security PDS to security personnel, and the region (read access only).
- Step 5. Ensure that the NPTABLES DD points to your security PDS.

**Note**

The NPTABLES DD in your RUNSYSIN member points to *dsnpref.ASvvv.ASTEXEC* by default.



# 4

---

## Starting and Stopping a Region

This chapter describes how to start and stop a region.

**This chapter contains the following topics:**

- Starting a Region
- Shutting Down a Region

---

## Starting a Region

To start a region, you need to run it as either a job or a started task. (If you follow the instructions in the *Installation and Setup Instructions*, you should have set up a started task.)

**Note**

To start a region on a VM system, log on to the SOLVE virtual machine.

Users log on to a region by using the user IDs and passwords specified in their UAMS (or external security package) records.

## Responding to WTOR Confirmation Message

If you have implemented region startup confirmation, the RMIWTO06 WTOR message is displayed and startup pauses. This may occur during system image load or SNA network discovery.

The WTOR message enables you to change the startup parameters. If a reply to the message is not made within a specified time, startup continues.

For information about startup confirmation, see the help for the AUTOIDS and AUTOSNACNTL parameter groups.

### Continue Startup With no Change

To continue startup with no change to the parameters, reply as follows:

```
R n,U
```

*n* is the identification number of the WTOR message.

## Continue Startup With Changes

To continue startup with changes to the parameters, reply as follows:

```
R n,parameter-1=value-1[,parameter-2=value-2[,...[,parameter-n=value-n]]]
```

Table 4-1 lists the parameters that you can use in your reply. It matches the parameters with the fields in the corresponding parameter group specification panels.

*Table 4-1. RMIWTO06 WTOR Message—Reply Parameters*

Parameter Name	Field Label
<b>System image load (AUTOIDS parameter group)</b>	
SYSTEM	System Image Name
VERSION	Version
MODE	Automation Mode
COLD	Cold Start on Next Restart?
<b>SNA network discovery (AUTOSNACNTL parameter group)</b>	
COLD	Cold Start on Next Discovery?
CMDS	VTAM Commands per Time Frame
TIME	Time Frame in Seconds

If you reply to change parameters, you will be asked to confirm your changes. You can then make additional changes or accept the displayed values.

### *Example*

The following reply changes the system image to be loaded to SOLVPROD version 2:

```
R n,SYSTEM=SOLVPROD,VERSION=2
```

---

## Shutting Down a Region

Shutdown commands are entered from the Message Monitor panel or from the Command Entry panel.

If you have the necessary authority, you can shut down the region by issuing the SHUTDOWN or FSTOP command.

For more information on the SHUTDOWN and FSTOP commands, see the *Command Reference* manual.

### SHUTDOWN Command

The SHUTDOWN command only stops the region when the last user logs off. When you issue the SHUTDOWN command, a broadcast is issued to all users. No further logons are accepted until the region is restarted, or the SHUTDOWN CANCEL command is issued.

Resource states are not affected by issue of the SHUTDOWN command.

The states of the resources managed by the region are not affected by the issue of the SHUTDOWN command.

### FSTOP Command

The FSTOP command immediately disconnects user sessions and shuts down the region, but does not affect the state of resources.

Use of the FSTOP command should be restricted, or the command totally disabled.

**Caution**

If you are running another product in the same region, it will also stop if the FSTOP command is issued.

# 5

---

## Administering a Multisystem Environment

This chapter describes how to establish and maintain your multisystem environment.

**This chapter contains the following topics:**

- Multisystem Operation
- Types of Data
- Linking Regions and Synchronizing Databases
- Keeping Track of Linked Regions
- Unlinking Regions
- Transmitting Records
- Developing and Testing a System Image in Isolation

---

## Multisystem Operation

Automation Services provides focal point management to support multisystem operation (that is, management at a focal point with subordinates feeding information to it) as follows:

- Peer-to-peer architecture—supports full connectivity between multiple regions. Regions linked in this way are known as focal point regions. (A stand-alone region is also regarded as a focal point region.)

When regions are communicating with each other, authorized users can monitor and control all managed resources from any terminal connected to any region.

- Subordinate—enables you to reduce the amount of traffic in your multisystem environment. You link subordinates to focal point regions that provide central monitoring and control. A subordinate has visibility and control of the locally managed resources only.

In a multisystem environment, each region can run independently of the other regions. If no communication links are available, each region still provides full monitoring, control, and automation of its own managed resources.

To link a focal point region to another focal point region, or to link a subordinate to a focal point region, you need to perform the procedure described in the section, *Performing the Link and Synchronize Procedure*, on page 5-10.

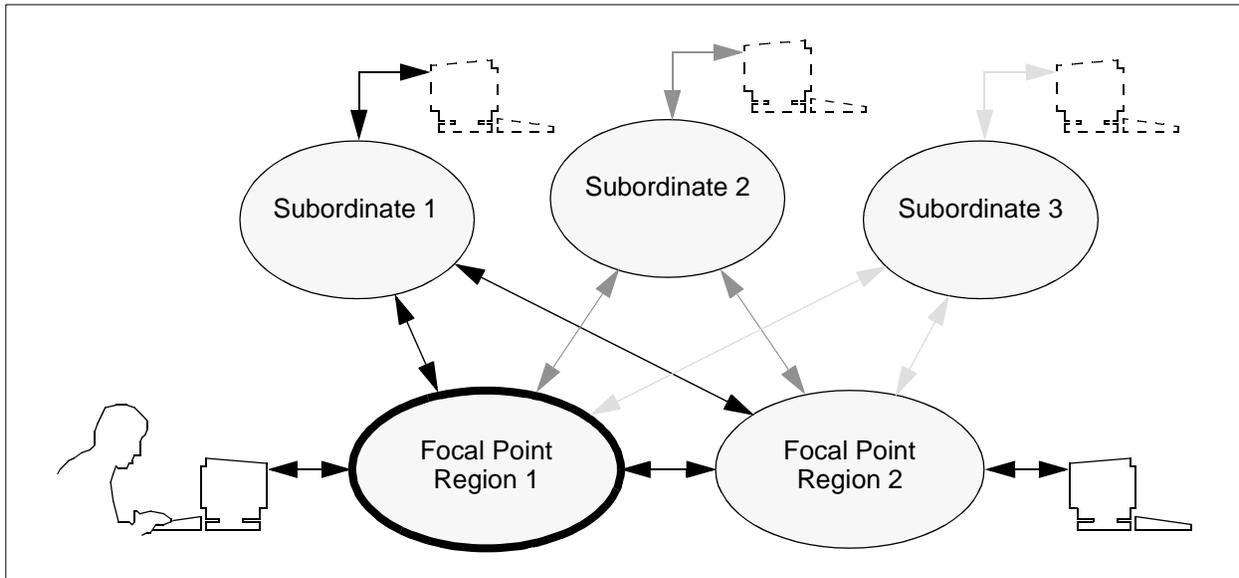
### Note

You can link as focal points only those regions that are licensed for the same products. For a subordinate-focal point link, the products licensed in the subordinate region can be a subset of the products licensed in the focal point region.

Figure 5-1 shows an example of a multisystem environment, where focal point region 1 is the main monitoring region and focal point region 2 is a secondary region that can serve as a backup. Note the following:

- A focal point region links to all other focal point regions and subordinates.
- A subordinate links to focal point regions but does not link to another subordinate.

Figure 5-1. Multisystem Operation



Automation Services offers multisystem support at the following levels:

- Service level, through the service (status) monitor
- Resource level, through resource (status) monitor and graphical monitor
- Event level, through the consolidated console and the alert monitor

This chapter describes how to set up multisystem support for products that use Automation Services. See also the chapter on console message consolidation in the *Automation Services Common User Guide* for information about multisystem support at the event level.

## Links in a Multisystem Environment

The link established between two regions in a multisystem environment is an INMC link. The link is used to pass knowledge base updates, status change notification, and other information between the two regions. The link can use any combination of the following communication access methods: EPS, VTAM, and TCP/IP.

### Note

Only regions on OS/390 systems support the EPS communication access method.

For each region, the communication access methods available to it are specified by the MULTISYS parameter group. By default, the group enables VTAM only. (To access the MULTISYS parameter group, enter the **/ICS** shortcut.)

### Note

When a region is linked in a multisystem environment, you cannot change the access methods in its MULTISYS parameter group without first unlinking the region.

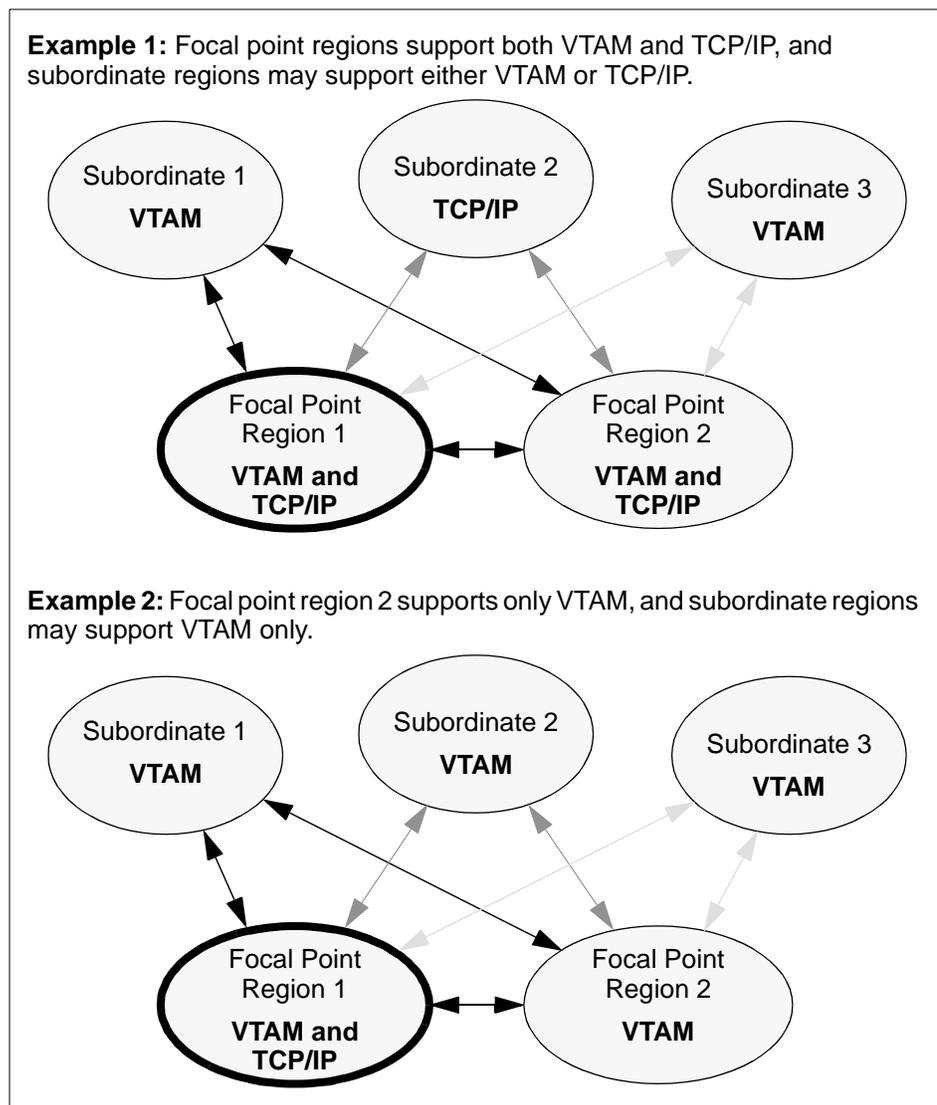
The INMC link between any two regions uses the access methods enabled by *both* regions (that is, the intersection of the two MULTISYS parameter groups). When multiple access methods are enabled, the link can use all these methods. This improves reliability because the link will function so long as one of the enabled methods is available.

When you plan your multisystem environment, you must ensure the following:

- All focal point regions must support at least one common type of access method.
- A subordinate region must support an access method that is also supported in all the focal point regions.

Figure 5-2 shows some examples.

Figure 5-2. Multisystem Link Configurations



## Multisystem Support in Sysplex

With the EPS access method, you can use the sysplex cross-system coupling facility (XCF) to implement your multisystem environment.

### Note

To support the EPS access method, an SSI region must be active in each of the cooperating systems and must be registered to XCF.

To register the SSI region to XCF, add the XCF=YES parameter to the *dsnpref.SIvvv.SSIPARM(SSIPARMS)* member.

## Multisystem Implementation Considerations

When you implement your multisystem environment, consider the following:

- Ensure that the link requirements are satisfied for the planned multisystem environment (see the section, *Links in a Multisystem Environment*, on page 5-4)
- When you link two regions, the knowledge base in one region overwrites the knowledge base in the other region.
- You can link a region to a focal point only. The focal point can be either a stand-alone region or part of a multisystem environment.
- You can only link a stand-alone region into a multisystem environment. You cannot link two multisystem environments together.

## Establishing Regions

When Automation Services is installed in your environment, two databases are downloaded. These databases, which can be tailored to suit your requirements, are:

- An icon panel database, where icon panel definitions are stored
- A knowledge base, where system image, resource, availability map, process, macro, command, and other definitions are stored (the RAMDB)

Populate these databases with definitions specific to your environment. These definitions may include the system image definitions for any other regions to be installed in your environment in the near future.

As you establish regions, link the new regions to the first region by using the Link Region and Synchronize Database option. (See the section, *Linking Regions and Synchronizing Databases*, on page 5-8). Once databases are linked, future synchronization is automatic. You can make changes to the database in one region and the changes are sent to the databases in the linked regions.

### Note

Synchronization does *not* apply to the NCL procedures represented by the registered commands and macros. Changes to these NCL procedures are *not* automatically reflected in the linked regions.

In a multisystem environment, you can monitor and control the resources in *all* linked regions from a single focal point. The types of data that can be passed between linked regions are described in the section, *Types of Data*, on page 5-7.

---

## Types of Data

There are several types of data that pass between regions. These data types are:

- Service status events—service status events are generated following the receipt of an external event, or message, that contains status change information for a resource that is a member of a service.
- Resource status events—resource status events are generated following the receipt of an external event, or message, that contains status change information for a specific resource. These events remain within the region that owns the resource and are only shipped to other regions when a status update is requested.
- Registered commands—registered command data flow occurs when an operator issues a registered command against a resource that is managed by a different region.
- Alerts—an alert notifies an operator that something has happened that might require manual intervention. Alerts and alert updates flow from the originating region to all linked focal point regions, where they are visible on the alert monitor.
- Database maintenance events—database maintenance events are generated after any knowledge base maintenance. These events inform remote regions of database updates, so that database synchronization can be maintained between linked regions.
- System commands and responses—this type of data includes the command responses that are sent from a remote region (where a system command was actually executed) to the local region (where the command was originally issued). The data is usually in the form of system messages, but this depends on the actual command.
- Unsolicited system message traffic—unsolicited system message traffic consists of all unsolicited system messages that flow between regions as a result of console consolidation. (Console consolidation is described in the *Automation Services Common User Guide*.)

---

## Linking Regions and Synchronizing Databases

When the first region is established in your environment, two databases are downloaded and can be customized for your environment. Together, these two databases (the Automation Services database and the icon panel library) form the knowledge base.

To build a multisystem environment, you start by linking two regions, then continue on to link in any other regions. The linking process also synchronizes the knowledge bases of these regions.

### Note

For linked focal point regions, synchronization is complete and the focal point knowledge bases duplicate each other.

For linked subordinates, synchronization is complete only to the extent of the relevant definitions in the knowledge base. For example, a subordinate knowledge base does not contain all system images. A subordinate knowledge base contains only those images that represent the environment the subordinate is managing.

When you link two regions, the local region in which you perform the link operation receives the knowledge base from the remote region you want to link to, which must be a focal point region. When you link a region into an existing multisystem environment, that region must be a stand-alone region.

During the linking and synchronization process, the knowledge base in the local region is overwritten by the knowledge base in the remote region.



### Warning

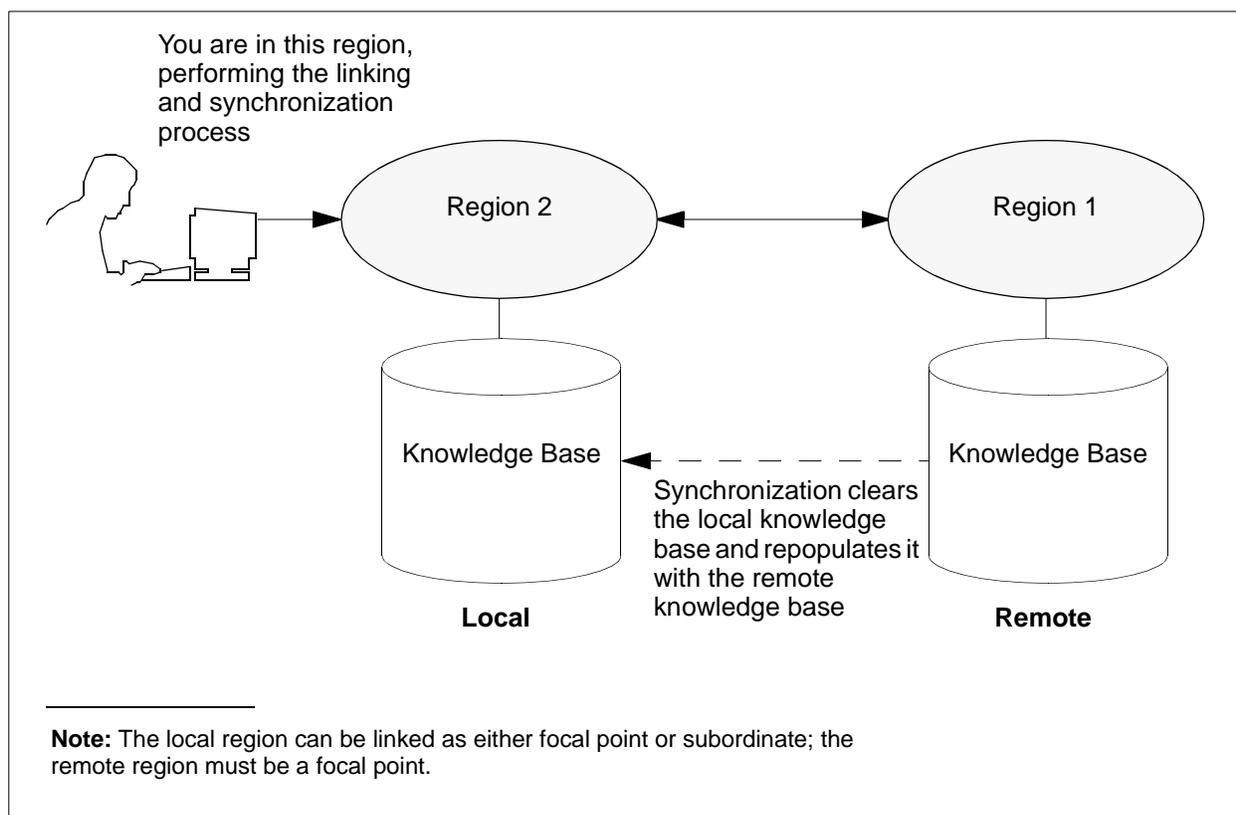
If the local knowledge base has been customized and contains definitions that you want to retain in the synchronized knowledge bases, you must transmit these definitions to the remote knowledge base before you link the regions. See the section, *Transmitting Records*, on page 5-20, for information about how to transmit definitions stored in the knowledge base.

### Note

If, for whatever reason, the local region terminates during the linking and synchronization process, the local knowledge base can become corrupted and you might not be able to restart the region. Replace the corrupted knowledge base with your backup. For information about how to back up a knowledge base, see the section, *Database Maintenance*, on page 6-3.

Figure 5-3 shows the link and synchronize operation.

Figure 5-3. Linking Regions



After you link the regions, the knowledge bases in them are synchronized and will remain synchronized. If you make changes to the knowledge base in one region, the changes are propagated to the other regions.

This section describes the procedures to perform if you want to link and synchronize other regions to this region.

## Performing the Link and Synchronize Procedure



### Warning

During the linking and synchronization process, the knowledge base in the local region is overwritten by the knowledge base in the remote region. If the local knowledge base has been customized and contains definitions that you want to retain in the synchronized knowledge bases, you must transmit these definitions to the remote knowledge base before you link the regions. See the section, *Transmitting Records*, on page 5-20, for information about how to transmit definitions stored in the knowledge base.

After you perform a link and synchronize procedure, the regions remain linked, and future database synchronization is automatic.

### Caution

Do not add, update, or delete database records in any linked regions while database synchronization is in progress. These changes might not be propagated to the new region.

The procedure for linking and synchronizing databases is as follows:

- Step 1. Log on to the region to be synchronized with the tailored (remote) region.
- Step 2. Enter the `=/MADMIN` shortcut to display the Multi-System Support Menu. An example is shown in Figure 5-4.

Figure 5-4. Multi-System Support Menu

```
SOLVPROD----- Automati      Menu -----/MUADMIN
Select Option ==>

  TI - Transmit System Image           TRANSYS
  TS - Transmit Services                TRANSER
  TC - Transmit Common Components      TRANCC
  TR - Transmit Ruleset                 TRANRUL
  LR - List Linked Regions              LISTREG
  SD - Link Region and Synchronize Database
  U  - Unlink Region DENML
  L  - View Task Log
  X  - Exit
```

Enter **SD** here to initiate the link and synchronize operation.

- Step 3. Select option **SD** to establish a link between the local region and another region, and to update the knowledge base of the local region.

A Remote System Identification panel is displayed.

Step 4. Supply the details for the operation as follows:

- a. Specify the ACB name (primary name) of the remote focal point region to which you want to link this region.

**Note**

The ACB name of a region is specified during the setup of the product (see the *Installation and Setup Instructions*). You can use the SHOW PARMS command to determine the name (see the online help).

- b. Specify whether the local region is to be a focal point region or a subordinate region. If it is to be a subordinate, specify the name of the system image that is to be used by it. For example, you can use the system ID in the SYSTEMID parameter group as the system image name.



**Key Concept**

Each subordinate is assigned a unique system image name, and it can use an image by that name only. When you build your resource management environment for a subordinate, you must build the environment under the system image name specified during the linking operation.

Focal points cannot use a system image that belongs to a subordinate.

For information about how to define a system image, see the *Automation Services Common User Guide*.

- c. Enter ***dsnpref.rname.RAMDB.WORK*** in the Work Dataset field, where *dsnpref* is the dataset name prefix you used during the installation of the software and *rname* is the name of the local region. Using this dataset speeds up processing significantly.

**Note**

On VM systems, prefix RAMDB.WORK by the high-level qualifier you specified during product setup.

- d. Specify the communication access methods to be used during synchronization. You can select any combination of the access methods; however, you can select an access method only if it is allowed in the MULTISYS parameter group.

Figure 5-5 shows an example of a completed panel.

Figure 5-5. Remote System Identification Panel

```
SOLVPROD----- Automation Services : Remote System Identification -----
Command ==>                                     Function=SYNCHRONISE

Identify the Remote Region:
  Primary Name ..... SOLV3           (Must be a FOCAL system)

Role in Multi-System Operation .. SUBORDINATE (Specify FOCAL or SUBORDINATE)
Subordinate System Image Name ... SOLVPROD

Work Dataset ..... ALSO0.SOLVPROD.RAMDB.WORK

Specify the Communication Access Methods to be used by this system:
  Use VTAM? ..... YES
  Use EPS? ..... NO
  TCP/IP Host Name/Addr ..... ALMVS3.AL.XYZ.COM
  Port Number ..... 2636
```

Step 5. After you have specified the details, press F6 (Action) to initiate the linking process.

A confirmation panel is displayed, as shown in Figure 5-6.

**Caution**

To cancel the synchronization procedure, press F12 (Cancel) now.

Figure 5-6. Confirm Database Synchronization Panel

```
SOLVPROD--- Automation Services : Confirm Database Synchronization -----

                SYNCHRONIZE Automation Services DATABASES

-----
WARNING: THIS OPTION WILL ERASE ALL DATA IN THE Automation Services
DATABASE AND REPLACE IT WITH DATA FROM THE NOMINATED REGION
-----

The Automation Services database includes:

* The resource definition database
* The ICOPANL panel library

PROCEDURES CURRENTLY ACCESSING THIS DATABASE WILL BE FLUSHED

Press Confirm key to confirm request, Cancel key to cancel the request

F1=Help      F2=Split      F9=Swap      F6=Confirm
F12=Cancel
```

Step 6. Press F6 (Confirm) to initiate region linking and knowledge base synchronization. A status panel, shown in Figure 5-7, is displayed.

Figure 5-7. Monitoring the Progress of Knowledge Base Synchronization

```

SOLVPROD----- Automation Services : Database Synchronization Status -----
Command ==>

  Status ..... ENDED                Task Number .... 0018
  Remote Primary Name ..... SOLV3    Started By ..... USER01
                                       Start Time ..... 13.52.52 07-JAN-1998
                                       End Time ..... 13.55.30 07-JAN-1998

Message

. Icon Panel Library -----
| Processed Successfully .. 1
| In Error ..... 0
|-----

. Automation Services Database -----
| Processed Successfully .. 647      Total Records ..... 647
| In Error ..... 0                  Percent Complete ..... 100
|
| 0                25                50                75                100
|-----
|
F1=Help      F2=Split      F3=Exit
              F9=Swap

```

You can press F3 (Exit) to exit the status panel at any time without affecting the link and synchronize procedure. If you exit early, note the task number for later reference.

## Monitoring the Synchronization Procedure

While the synchronization procedure is in progress, the Synchronize Database Status panel is refreshed automatically every 10 seconds, keeping you informed of progress. This panel can be refreshed manually at any time by pressing the ENTER key.

If you exited the status panel and want to check the status of the synchronization, proceed as follows:

- Step 1. From the Multi-System Support Menu, select option **L** to view the administration task log. An example is shown in Figure 5-8.
- Step 2. Enter **S** beside the appropriate entry from the log to view the status of the task.

The Administration Task Log contains up to 50 entries at any given time. Each task is allocated a sequential task number (between 1 and 50) as it commences. When the maximum task number is reached, allocation restarts from one and the oldest status records are overwritten. To delete a completed or failed task from the log, apply the **D** (Delete) action.

*Figure 5-8. Browsing the Task Log*

```
SOLVPROD-----Automation Services : Administration Task Log -----
Command ==>                               Scroll ==> PAGE

Task ACB Name Type      Status  % Done Start Time      End Time
0019 SOLV3  SYNC    RUNNING  50    13.52 07-FEB-1998
0018 SOLV3  SYNC    ENDED    100    13.52 07-JAN-1998  13.55 07-JAN-1988
0017 SOLV9  TRAN-CMP ENDED    100    15.44 06-JAN-1998  15.45 06-JAN-1998
0015 SOLV1  TRAN-CMP ENDED    100    13.53 05-JAN-1998  13.54 05-JAN-1998
0014 SOLV11 SYNC     ENDED    100    10.02 31-DEC-1997  10.03 31-DEC-1997
**END**

S=Select D=Delete

F1=Help      F2=Split    F3=Exit      F5=Find      F6=Refresh
F7=Backward F8=Forward  F9=Swap
```

## Background User Considerations

When you establish a region, a UAMS background system (BSYS) user ID for that region is automatically defined. The background user ID comprises the 4-byte region domain ID, followed by the characters BSYS. For fully functioning communication links to be established between regions, the BSYS user ID of each region must be duplicated in each linked region.

During a link and synchronize procedure, any required BSYS user IDs are defined automatically, provided that the following conditions apply:

- You have UAMS maintenance authority on all the linked regions.
- The linked regions are all working when the request is made.

If either of these conditions does not apply, then any required BSYS user IDs must be defined manually. The simplest way to do this is to copy the BSYS user ID for the current region from the UAMS User Definition List and update the user ID. (To access the UAMS maintenance functions, enter the `/UAMS` shortcut.)

The link and synchronize request is rejected if both of the following apply:

- You do not have UAMS maintenance authority in either the local or the remote region. (The user ID of the person who requests the link and synchronize procedure must be defined in both the local and remote regions.)
- The required BSYS user IDs are not defined in the local or the remote region.

After you perform a link and synchronize procedure, some required BSYS user IDs might not have been defined (due to regions not being active at the time or to you not having UAMS maintenance authority on all regions). Although a theoretical link exists, a true communications link has not been established between those regions that do not contain the requisite BSYS user IDs. Any subsequent updates to those regions will remain on the staging file until the necessary BSYS user IDs are defined.

## Knowledge Base Synchronization Maintenance

Automation Services maintains synchronization between linked knowledge bases by using a staging file.

When a knowledge base update occurs, details of the update are stored in the staging file as follows:

- For an update in a focal point region, a separate update record is written for each affected linked region.
- For an update in a subordinate region, a single update record is written for a linked focal point region.

A record stays in the staging file until the update is performed successfully in the destined region. If the region is inactive, the record stays in the staging file until the region is started.



### Warning

If the staging file becomes full, knowledge base synchronization cannot be maintained and the local region is unlinked automatically. A staging file can become full if a linked region remains inactive for an extended period of time. If an extended downtime is planned for a linked region, unlink the region before inactivation.

---

## Keeping Track of Linked Regions

Links between regions are established automatically when you synchronize databases.

Automation Services maintains the list of linked regions in your multisystem environment. To list them, enter the **/LISTREG** shortcut.

The Linked Regions list panel displays the ACB names and brief descriptions of linked regions (see Figure 5-9). It also displays the status of the data flow traffic managers. Press F11 (Right) to scroll right to display more information.

Figure 5-9. Linked Regions Panel

```

SOLVPROD----- Automation Services : Linked Regions -----
Command ==>                                         Scroll ==> 10
                                                    S/B=Browse U=Update

Primary
Name      Role      Link Name  Short Description
SOLV13   FOCAL      SOLV13     Link Definition for Region SOLV13
SOLV14   FOCAL      SOLV14     Link Definition for Region SOLV14
SOLV15   SUBORDINATE SOLV15     Link Definition for Region SOLV15
**END**

```

```

SOLVPROD----- Automation Services : Linked Regions -----
Command ==>                                         Scroll ==> PAGE
                                                    S/B=Browse U=Update

Primary      System
Name      Role  Name  Vers  SMFID  Prefix  Isolated?
SOLV13   FOCAL SOLV13  0001  AL01   S13
SOLV14   FOCAL SOLV14  0001  AL02   S14
SOLV15   SUBOR SOLV15  0001  AL03   S15

```

```

SOLVPROD----- Automation Services : Linked Regions -----
Command ==>                                         Scroll ==> PAGE
                                                    S/B=Browse U=Update

Primary      System      Data      Resource Service
Name      Role  Name  Vers  Traffic Traffic  Event  Event  Event  File
SOLV13   FOCAL SOLV13  0001  ACTIVE ACTIVE  ACTIVE - - -
SOLV14   FOCAL SOLV14  0001  - - - - - - -
SOLV15   SUBOR SOLV15  0001  ACTIVE ACTIVE  - - - -

```

```

SOLVPROD----- Automation Services : Linked Regions -----
Command ==>                                         Scroll ==> PAGE
                                                    S/B=Browse U=Update

Primary      System      ----- T C P / I P -----
Name      Role  Name  Vers  VTAM  EPS  TCP/IP  Port  Host Name/Address
SOLV13   FOCAL SOLV13  0001  YES  NO  YES  2613  ALMVS1.AL.STATE.COM
SOLV14   FOCAL SOLV14  0001  YES  NO  YES  2614  ALMVS2.AL.STATE.COM
SOLV15   SUBOR SOLV15  0001  NO   NO  YES  2623  ALMVS3.AL.STATE.COM

```

```

SOLVPROD----- Automation Services : Linked Regions -----
Command ==>                                         Scroll ==> PAGE
                                                    S/B=Browse U=Update

Primary      System
Name      Role  Name  Vers  Created      Last Updated
SOLV13   FOCAL SOLV13  0001  14-APR-1998 29-MAY-1998 15.33 SOL13BSYS
SOLV14   FOCAL SOLV14  0001  16-APR-1998 21-AUG-1998 10.41 SOL14BSYS
SOLV15   SUBOR SOLV15  0001  28-AUG-1998 31-AUG-1998 10.52 SOL15BSYS
**END**

```

You can apply the following actions to listed items:

- **B** to browse a link description (see Figure 5-10)
- **U** to update a link description or a (region identifier) message prefix

Figure 5-10. Browsing a System Link Definition

```
SOLVPROD----- Automation Services : Link Record Details -----
Command ==>                                     Function=Browse

Primary Name ..... SOLV13           Link Name ..... ALOS390
                                           Message Prefix... SV13A
Short Description ... Link Definition for Region SOLV13
Long Description ... This is a SOLVE:Operations Automation region.

. System Details -----
| System Name ..... SOLV13           Version ..... 0001
| Role in Multi-System ... FOCAL     Subordinate Image ..... -
| SMFID ..... AL01
| Access Methods: VTAM? .. YES       EPS? ... NO           TCP/IP? ... YES
|   TCP/IP Port ..... 2613
|   Host Name/Addr ..... ALOS3901.AL.STATE.COM
+ Traffic Status -----
| Data ..... ACTIVE
| Resource Events ..... ACTIVE       Service Events ..... ACTIVE
| Shared Events ..... -             File Transfer Events .... -
|-----|
F1=Help      F2=Split      F3=Exit      F4=Edit
                F9=Swap
```

## Unlinking Regions

You might want to unlink a region from the other regions in a multisystem environment (for example, for maintenance purposes). If a region is no longer of use and you want to remove it from the system, make sure you unlink it first. An unlinked region becomes a stand-alone region.

Unlink a region as follows:

Step 1. Log on to the region you want to unlink.

Step 2. Enter the **/MADMIN.U** path.

The Confirm Unlink Panel is displayed (as shown in Figure 5-11).

### Caution

To cancel the unlinking procedure, press F12 (Cancel) now.

Figure 5-11. Confirming a Region Unlink Operation

```
SOLVPROD----- Automation Services : Confirm Unlink -----NET001
Command ==>

                                CONFIRM UNLINK OF DENM13

-----
WARNING: The unlinking of DENM13 will result in its ISOLATION
         from all other regions.

Once unlinked you will not be able to monitor or control another
region's resources from this region and you will not be able to
monitor or control this region's resources from any other region.

To reconnect this region to another Automation Services region
after unlinking you will need to perform database synchronization.
-----

Press the Confirm key to unlink.
Press the Cancel Key to cancel the request.

F1=Help      F2=      F6=Confirm
              Press F6 to unlink the region.      F12=Cancel
```

Step 3. Press F6 (Confirm) to proceed with the unlinking procedure.

To relink a region, synchronize that region with one of the regions in the multisystem environment. See the section, *Linking Regions and Synchronizing Databases*, on page 5-8 for this procedure.

---

## Transmitting Records

You can transmit (that is, copy) knowledge base records from the local region to a remote region that is *not linked* to it.

You *cannot* transmit a system image to a region in which the image is currently loaded. You *cannot* transmit and replace a rule set when the rule set is currently loaded in the remote (target) region.

You can transmit the following record types:

- System image definitions
- System image component definitions
- Service definitions
- Rule sets
- Common components such as command definitions and consolidated console profiles

By specifying the appropriate transmission mode on the Remote System Identification panel, you can specify how to update the records in the remote region.

## Transmission Modes

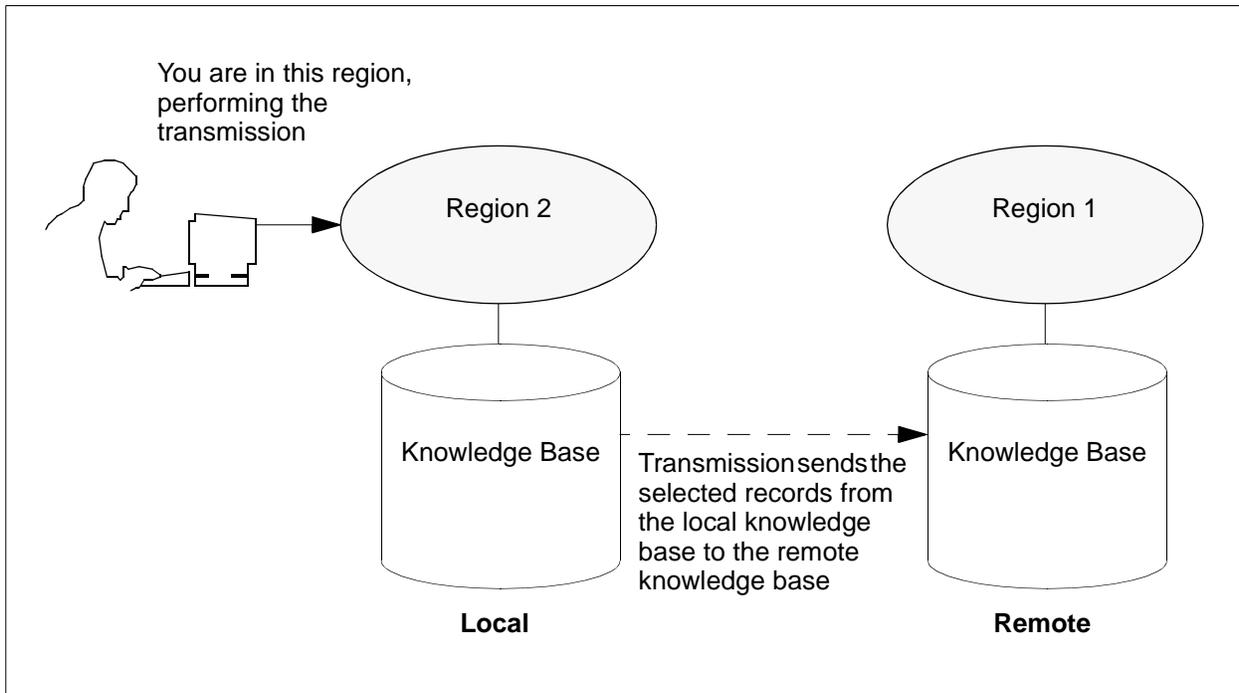
The available transmission modes are as follows:

- Replace (R) deletes any existing remote records, then transmits the local records.
- Overlay (O) replaces existing remote records with the same name, adds records that do not already exist, but does not delete any records in the remote knowledge base.
- Merge (M) adds records that do not already exist, but does not have any affect on existing records in the remote knowledge base.

## Transmission Procedure

Figure 5-3 shows the transmit operation.

Figure 5-12. Transmitting Knowledge Base Records



Transmit knowledge base records as follows:

- Step 1. Log on to the region from which you want to transmit the records.
- Step 2. Enter the **/MADMIN** shortcut to display the Multi-System Support Menu.
- Step 3. Specify the option you require in the Select Option ==> field.
- Step 4. Press ENTER to display a Remote System Identification panel.
- Step 5. Specify the ACB name (primary name) of the region to which you want to transmit records.

Step 6. If you specified the TI, TS, or TR option, continue with this step. If you specified any other transmission options, go to the next step:

<b>If you want to select the ...</b>	<b>Complete the ...</b>
TI - Transmit System Image option	System Name and Version fields.
TS - Transmit Services option	Version field.
TR - Transmit Ruleset option	Ruleset Name field.

Step 7. Specify the transmission mode as follows:

<b>If you want to ...</b>	<b>Specify ...</b>
Replace a set of records, such as a system image, a rule set, <i>all services</i> , or <i>all elements</i> of a component	<b>REPLACE</b> in the Transmission Mode field.
Update a region by adding new records, without updating existing records	<b>MERGE</b> in the Transmission Mode field.
Update a region by adding new records <i>and</i> updating existing records	<b>OVERLAY</b> in the Transmission Mode field.

Step 8. Specify the communication access methods to be used for transmitting the selected records. You can enable any combination of the access methods.

Step 9. Press F6 (Action) to select the specified option. What you do next depends on the specified option and transmission mode.

Step 10. If a selection list is displayed, continue with this step. If the Confirm Transmit panel is displayed, go to Step 12.

**If you selected ... And the transmission mode is ... Then a ...**

Option TC - Transmit Common Components	REPLACE	List of all the common component categories is displayed.  To select all definitions in a category for transmission, enter <b>S</b> (select) beside it.
	MERGE or OVERLAY	List of all the common component categories is displayed.  To select all definitions in a category for transmission, enter <b>S</b> (Select) beside it.  To select specific definitions in a category for transmission, enter <b>L</b> (List) beside the category to list the definitions, then enter <b>S</b> (Select) beside the definitions to be transmitted. After you have made your selections, press F3 (Exit) to return to the Common Component List panel.
Other transmission options	MERGE or OVERLAY	Selection list is displayed.  To select all definitions for transmission, press F4 (All).  To select specific definitions for transmission, enter <b>S</b> (Select) beside the definitions to be transmitted.

Step 11. When you have selected all the definitions you want to transmit, press F6 (Transmit) to request transmission.

A Confirm Transmit panel (displaying the transmission details you entered) is displayed before transmission is initiated. At this point, you have the option of confirming or cancelling your transmission request.

Step 12. Press ENTER to confirm transmission or F12 (Cancel) to cancel the transmission.

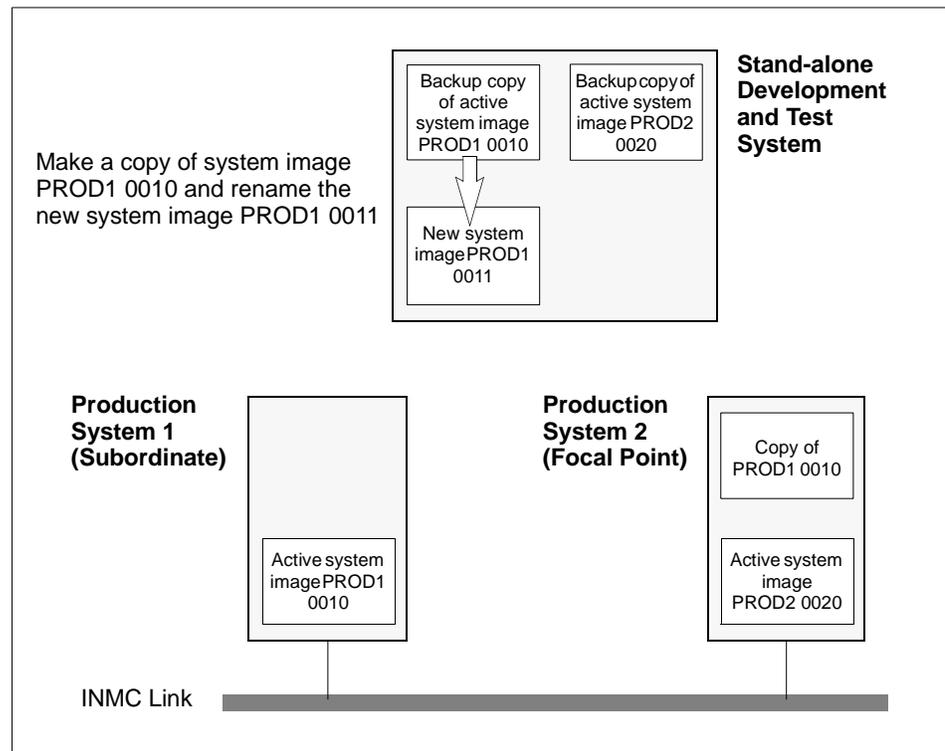
If you press ENTER to confirm transmission, a status panel is displayed, showing the progress of the transmission. If you choose to exit the status panel, you can check the status of the task subsequently by viewing the administration task log. Before you exit, note the task number for future reference. (See the section, *Monitoring the Synchronization Procedure*, on page 5-14 for information about the administration task log.)

If you press F12 (Cancel) to cancel the transmission, you are returned to the Multi-System Support menu.

## Developing and Testing a System Image in Isolation

A typical multisystem setup is likely to consist of a group of linked production systems (or regions), and a stand-alone development and test system, as shown in Figure 5-13.

Figure 5-13. Redeveloping a System Image in a Stand-alone Region



The development and test system should have copies of the active system images of all your production systems.

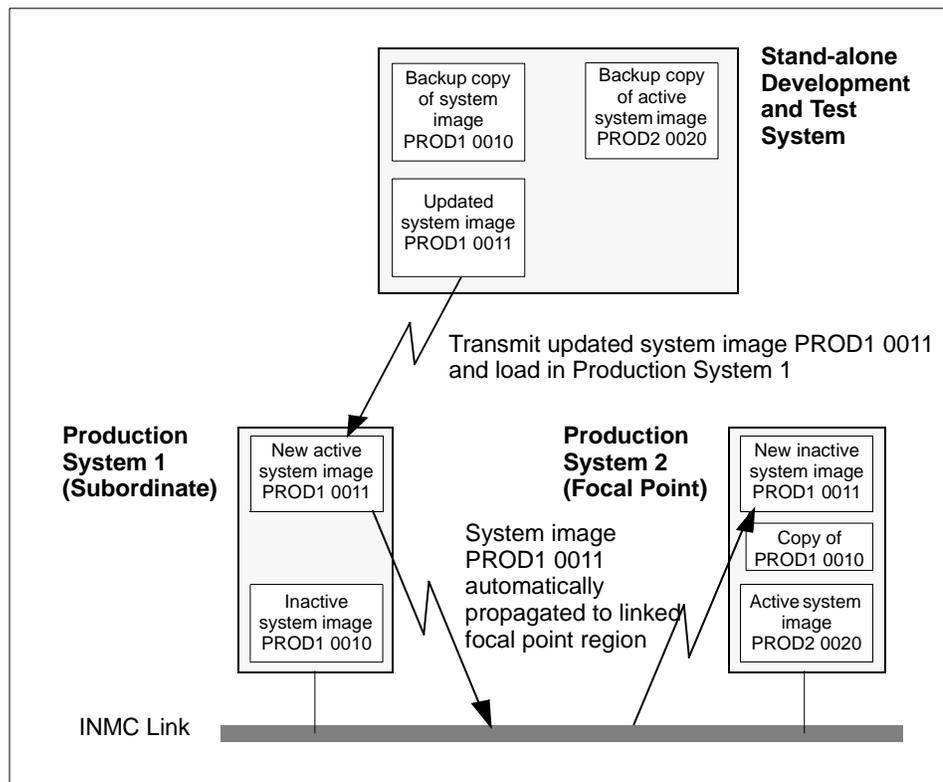
To update and test a current system image in a stand-alone region, you do the following:

- Step 1. Transmit a copy of the active system image that you want to update to the stand-alone development region.
- Step 2. Copy this system image to a new version.
- Step 3. Make the required changes to the new version of the system image and test these changes.
- Step 4. When you are satisfied with the updates you have made, transmit the updated system image to the production region where it originated (as shown in Figure 5-14).

The new system image is automatically copied to all affected linked regions.

- Step 5. To replace the currently active system image with the updated version, load the new system image.

Figure 5-14. Updating the Active System Image





---

## Maintaining Automation Services

This chapter describes maintenance tasks.

**This chapter contains the following topics:**

- Changing the Activity Log
- Database Maintenance

---

## Changing the Activity Log

The activity log records all the activities that occur within a region. By completing the Log File specifications on the first panel of the LOGFILES parameter group, you can specify whether the activity logs specified in the JCL are to wrap or not.

If you specify Yes (the default), Automation Services writes to the defined logs in a cyclic sequence. When the current activity log is full, Automation Services starts logging to the next defined log, and so on. When all logs are full, the first log is cleared and reused. If you only define one log, Automation Services clears that log each time it is full and continues logging to the same log. (To define additional logs, update the LOGFILES parameter group.)

Logs can be swapped manually, for example, to separate system maintenance activities from records logged during normal operation.

To change logs manually, complete these steps:

- Step 1. Enter **/LOGSWAP** to clear the next log in the sequence and start logging to the new log.

A confirmation panel, shown in Figure 6-1, is displayed.

- Step 2. Press ENTER to confirm the log swap.

*Figure 6-1. Confirming a Log Swap*

```
SOLVPROD----- Activity Log Services : Confirm Swap Log -----
Command ==>>

                               SWAP ACTIVITY LOG
-----
| This action will swap activity logging to the next defined VSAM log file.
| The new log will be emptied before swapping.
| If only one log file is available, it will be emptied and re-used.
| Any procedure that is currently accessing the log file may be flushed by
| this operation.
-----

Press the Action key to request the log swap, or the Cancel key to cancel it.

F1=Help      F2=Split      F9=Swap      F6=Action
F12=Cancel
```

---

## Database Maintenance

The method you select for performing database maintenance depends on the configuration of your regions and your operations requirements.

Backup methods depend on whether your regions are:

- Unlinked, non-production regions
- Unlinked production regions
- Linked production and non-production regions

### Unlinked Non-production Regions

You can back up the database to tape, as follows:

- Step 1. Stop the region.
- Step 2. Back up the database.
- Step 3. Restart the region.

## Unlinked Production Regions

If the region cannot be shut down for database backup, then the suggested procedure for backing up unlinked production regions is as follows:

- Step 1. Create a duplicate region with its own knowledge base.

**Caution**

Ensure that the databases of the duplicate region are not on the same DASD as the production region databases.

**Note**

If the region supports shared system images (for example, a region licensed for the SOLVE:Operations Automation product), create the duplicate on a different system. You cannot link regions that support shared system images if those regions are on the same system.

- Step 2. Use the Link Region and Synchronize Database option (on the Multi-System Support Menu) to link the duplicate region to the production region. This copies the databases of the production region to the duplicate region.

- Step 3. In the duplicate region:

- a. Switch off console consolidation in the CCONSOLIDATN ICS parameter group if applicable.
- b. Load an empty system image with no EventView rule set.

**Note**

The duplicate region does not perform any automation, it just contains an up-to-date, mirror image of the production databases.

- Step 4. When backup is required, stop the duplicate region and perform the backup to tape.

Any database updates that occur while the duplicate region is stopped are held by the staging file. The updates are sent to the duplicate region when it is restarted.

## Linked Regions

Each region contains a copy of the database of each linked region. Updates to any database are immediately propagated to the databases of each linked region.

### Backups to Tape When Regions Cannot Be Shut Down

If backups to tape are required, and no linked regions can be shut down, do the following:

- Step 1. Create a duplicate region, and link it to the production region (as described in the previous section, *Unlinked Production Regions*).
- Step 2. Stop the duplicate region, and perform the backup to tape.

Any database updates that occur while the duplicate region is stopped are held by the staging file. The updates are sent to the duplicate region when it is restarted.



---

## Customizing the Display Attribute Tables

This chapter describes how to customize the display attributes of the monitors.

**This chapter contains the following topics:**

- What Is a Display Attribute Table?
- Editing a Display Attribute Table
- The Logical State Attributes Table
- The Automated Mode Attributes Table
- The Manual Mode Attributes Table
- The SNA Resource Logical State Normalization Table

---

## What Is a Display Attribute Table?

The display attribute tables list the following:

- The range of possible logical states for resources and services
- The display attributes that are used by the status and graphical monitors to show the logical state of resources and services

This chapter describes how to customize the four display attribute tables:

- The Logical State Attributes Table
- The Automated Mode Attributes Table
- The Manual Mode Attributes Table
- The SNA Resource Logical State Normalization Table

---

## Editing a Display Attribute Table

To edit and customize the information in these tables, complete these steps:

- Step 1. Enter **/ASADMIN** on the primary menu to display the Administration Menu.
- Step 2. Select option **A** - Display Attribute Tables to display the list of attribute tables.
- Step 3. Select the table that you want to edit.
- Step 4. Press **F4** to edit the display attributes.
- Step 5. Press **F3** to save your changes and return to the Customization Menu.

### Caution

When you make a change to a display attribute table, this change is not reflected immediately. Changes are applied to individual resources and services when an event arrives that forces an update of the resource or service status. This can result in a monitor displaying a mixture of old and new attributes. To avoid this situation, issue the **CHECKALL** command from the resource or service monitor immediately after making changes to the display attributes tables. This forces the generation of events for all resources and services, and all status displays are updated.

## The Logical State Attributes Table

The Logical State Attributes Table displays the attributes used by the status and graphical monitors to show the logical states of resources and services. The logical state of a resource or service is derived from the values defined in the Automated and Manual Mode display attribute tables.

The display attributes used in this table are:

- Rank in severity
- Intensity
- Color
- Highlighting

Figure 7-1 shows a sample logical state attributes table.

Figure 7-1. *The Logical State Attributes Table*

```
SOLVPROD---- Automation Services : Logical State Attributes Table --$$ATTR-0000
Command ==>                                     Function=BROWSE
```

STATE	RANK	INTENSITY	COLOR	HIGHLIGHT
UNKNOWN	001	LOW	WHITE	NONE
INERROR	002	HIGH	WHITE	REVERSE
FAILED	003	LOW	RED	NONE
ATTENTION	004	LOW	PINK	NONE
DEGRADED	005	LOW	YELLOW	NONE
PENDING	006	LOW	TURQUOISE	NONE
STARTING	007	LOW	BLUE	NONE
STOPPING	008	LOW	BLUE	NONE
OK	009	LOW	GREEN	NONE

For example, if you want resources with a logical state of INERROR to be displayed in red and reverse video, at high intensity, you would assign the following values:

Intensity: HIGH  
Color: RED  
Highlight: REVERSE

### Note

Automation Services uses the blinking highlight attribute to display a resource when an automated action is being performed on the resource. It is therefore recommended that you do *not* use the BLINK highlight attribute.

## Logical State Attributes and SNA Network Summary Display

If you are using the NetMaster Network Automation product, color coding of the summary display also corresponds to the logical state attributes table. ACTIVE and INACTIVE, which do not appear in the table, assume the color of the OK state.

---

### The Automated Mode Attributes Table

The Automated Mode Attributes Table lists the logical states assigned to resources and services in Automated mode. The logical state is based on a combination of the mode, the desired state, and the actual state. The valid logical states are listed in the Logical State Attributes Table.

### Setting the Logical States for the Automated Mode

Set the logical states to best describe the situation. For example, if the actual state of a resource is INACTIVE and the specified desired state is ACTIVE, then in the automated mode of operation, Automation Services will try to bring the actual state to ACTIVE also. Use the PENDING logical state to indicate that the resource is in the process of becoming active.

Using the settings shown in Figure 7-1 on page 7-3 as an example, a resource that is PENDING is displayed in low intensity turquoise.

Figure 7-2 shows a sample Automated Mode Attributes Table.

Figure 7-2. The Automated Mode Attributes Table

```
SOLVPROD---- Automation Services : Automated Mode Attributes Table -$$ATTR-0000
Command ==>>>                                     Function=BROWSE
```

ACTUAL STATE	DESIRED STATE	
	ACTIVE	INACTIVE
ACTIVE	OK	PENDING
STARTING	OK	INERROR
STOPPING	INERROR	OK
DEGRADED	DEGRADED	OK
INACTIVE	PENDING	OK
FAILED	FAILED	INERROR
UNKNOWN	UNKNOWN	UNKNOWN

---

## The Manual Mode Attributes Table

The Manual Mode Attributes Table lists the logical states assigned to resources and services in Manual mode. The logical state is based on a combination of the mode, the desired state, and the actual state. The valid logical states are listed in the Logical State Attributes Table.

### Setting the Logical States for the Manual Mode

Set the logical states to best describe the situation. For example, if the actual state of a resource is INACTIVE, and the specified desired state is ACTIVE, then in the manual mode of operation the operator needs to take action. Automation will *not* take place to bring the actual state of the resource to ACTIVE (to satisfy the desired state) without operator intervention. Therefore, use the ATTENTION logical state to indicate to the operator that action is required.

Using the settings in Figure 7-1 on page 7-3 as an example, a resource that is in a state of ATTENTION is displayed in low intensity pink.

Figure 7-3 shows a sample Manual Mode Attributes Table.

*Figure 7-3. The Manual Mode Attributes Table*

```
SOLVPROD----- Automation Services : Manual Mode Attributes Table ---$$ATTR-0000
Command ==>                                     Function=BROWSE
```

ACTUAL STATE	DESIRED STATE	
	ACTIVE	INACTIVE
ACTIVE	OK	ATTENTION
STARTING	OK	ATTENTION
STOPPING	ATTENTION	OK
DEGRADED	ATTENTION	ATTENTION
INACTIVE	ATTENTION	OK
FAILED	ATTENTION	ATTENTION
UNKNOWN	UNKNOWN	UNKNOWN

```
F1=Help      F2=Split    F3=Exit     F4=Edit
F7=Backward  F9=Swap     F11=Panels
```

## The SNA Resource Logical State Normalization Table

The SNA Resource Logical State Normalization Table lists the logical states assigned to individual SNA resources. The logical state is based on a combination of the desired state and the actual state. The logical state for each SNA resource can be either OK or NOTOK.

This state normalization table is only used by SNA groups and only when the SNA group has a threshold specified for calculating the logical state of the SNA group. The table allows the resources within the group to be classified into two logical states: OK and NOTOK and the threshold specified is then used to assign the logical state of the SNA group.

### Note

This table does not affect the normal operation of SNA resources that are not in an SNA group.

## Setting the Logical States

Set the logical states to best describe the situation. For example, if the actual state of a resource is INACTIVE, and the specified desired state is ACTIVE, then the logical state of the SNA resource is NOTOK.

Figure 7-3 shows a sample SNA Resource Logical State Normalization Table.

Figure 7-4. The SNA Resource Logical State Normalization Table

```
--- Automation Services : SNA Resource Logical State Normalization Table -----
Command ==>>>                                     Function=BROWSE
```

SNA RESOURCE ACTUAL STATE	SNA RESOURCE DESIRED STATE	
	ACTIVE	INACTIVE
ACTIVE	OK	NOTOK
STARTING	OK	NOTOK
STOPPING	NOTOK	OK
DEGRADED	OK	NOTOK
INACTIVE	NOTOK	OK
FAILED	NOTOK	NOTOK
UNKNOWN	NOTOK	NOTOK

F1=Help      F2=Split      F3=Exit      F4=Edit  
F7=Backward      F9=Swap      F11=Panels

# 8

---

## Customizing Display Formats

This chapter describes how to customize the display format of the status and alert monitors.

**This chapter contains the following topics:**

- What Is a Status Monitor Display Format?
- Defining Status Monitor Display Formats
- What Is an Alert Monitor Display Format?
- Defining Alert Monitor Display Formats

## What Is a Status Monitor Display Format?

A status monitor display format determines what information is displayed for managed services or resources on the status monitor.

The status monitor can display up to five services and resources (five columns) across a single screen. Figure 8-1 shows a 1-column display, while Figure 8-2 shows a 2-column display.

Figure 8-1. Single-column Status Monitor Display

```

S=Status L=Log D=Display A=Act T=Term DB=Database ?=List Cnds
System Class Resource Desired Actual Mode Logical Ovr
SOLV INIT 1 ACTIVE CLASS=A JOBNAME=SANM11
SOLV INIT 2 ACTIVE CLASS=A JOBNAME=PICHULT
SOLV INIT 3 ACTIVE CLASS=A JOBNAME=PICHULT

```

Figure 8-2. Two-column Status Monitor Display

```

RMST0007 DISPLAY COLUMNS SET TO (2)
S=Status L=Log D=Display A=Act T=Term DB=Database ?=List Cnds
System Name Actual Logical System Name Actual Logical
SOLV 1 ACTIVE OK SOLV 2 ACTIVE OK
SOLV 3 ACTIVE OK SOLV 4 ACTIVE OK
SOLV 5 ACTIVE OK SOLV 6 ACTIVE OK
SOLV 7 ACTIVE OK SOLV 8 ACTIVE OK
SOLV 9 ACTIVE OK SOLV 10 ACTIVE OK
SOLV 11 ACTIVE OK SOLV 12 ACTIVE OK
SOLV 13 ACTIVE OK SOLV 14 ACTIVE OK
SOLV 15 ACTIVE OK SOLV JES2 DEGRADED ATTENTI
SOLV LINE1 ACTIVE OK SOLV LINE2 ACTIVE OK
SOLV LOGON1 ACTIVE OK SOLV PRT1 ACTIVE OK
SOLV SPOOL ACTIVE OK SOLV APPC ACTIVE OK
SOLV ASCH ACTIVE OK SOLV BACKUP ACTIVE OK
SOLV BACKWKLY ACTIVE OK SOLV BCKWK1 ACTIVE OK
SOLV BKNEW ACTIVE OK SOLV BSMVS1 INACTIVE OK
SOLV BSMVS4 INACTIVE OK SOLV CMNM1 INACTIVE OK
SOLV CMNM2 INACTIVE OK SOLV CMNM3 INACTIVE OK
SOLV CMNM4 INACTIVE OK SOLV CMNM5 INACTIVE OK

```

In the 1-column display, each line shows one managed item. In the 2-column display, each line shows two managed items. In the maximum 5-column display, each line shows the status of five managed items.

The information shown in each display column is determined by the applied status monitor display format. Default formats are supplied for each of the column displays, but you can define your own formats.

Multicolumn displays do not apply to status monitors with views.



**Key Concept**

A view customizes your status monitor for the specific purpose of monitoring certain classes of resources. Each view has associated with it a selected set of display formats.

---

## Defining Status Monitor Display Formats

Define status monitor display formats from the List Definition List panel. To access the list, enter the /ASADMIN.L shortcut.

Figure 8-3 shows an example. The Userid column identifies the type of display to which a format belongs. COLUMNS $n$  indicates that the format is for an  $n$ -column display.

Figure 8-3. List of Status Monitor Display Formats

```
SOLVPROD----- CAS : List Definition List -----
Command ==>>                                     Scroll ==>> CSR

                               S/B=Browse U=Update D=Delete C=Copy
Appl Typ  Userid  Name      Description                               File ID
$RM  PRI  COLUMNS1  DEFAULT  Default Status Monitor Format             MODSDIS
$RM  PRI  COLUMNS1  EXTDISP  Emphasise Extended Display Field         MODSDIS
$RM  PRI  COLUMNS1  LONGNAME Status Monitor - Long Rsc Names          MODSDIS
$RM  PRI  COLUMNS1  SYSID    Shows Current System for ALL Rsc        MODSDIS
$RM  PRI  COLUMNS1  SYSTEM  Shows Shared Rsc Current System         MODSDIS
$RM  PRI  COLUMNS2  DEFAULT  Default Status Monitor Format             MODSDIS
$RM  PRI  COLUMNS3  DEFAULT  Default Status Monitor Format             MODSDIS
$RM  PRI  COLUMNS4  DEFAULT  Default Status Monitor Format             MODSDIS
$RM  PRI  COLUMNS5  DEFAULT  Default Status Monitor Format             MODSDIS
```

The status monitor display formats are maintained as Common Application Services (CAS) lists. However, you *cannot* use the CAS application program interface to display the information specified by the display formats. (If you want to learn more about CAS lists, see the *Management Services Managed Object Development Services Programming and Administration Guide*.)

## Creating a Status Monitor Display Format

Create a status monitor display format from the List Definition List panel as follows:

- Step 1. Press F4 (Add). You can also use the C action code to open a copy of an existing display format definition that you can modify.
- Step 2. On the displayed List Description panel, fill in the following fields:

Field	Action
Appl ID	Ensure that the value is \$RM.
List Type	Ensure that the value is PRIVATE.
Userid	<p>Append a number from 1 through 5 to COLUMNS to identify the column display for which the format is created (for example, COLUMNS1 for a 1-column display). (COLUMNS<math>n</math> is the user that owns the format.)</p> <p>For a status monitor with the following views, specify the appropriate view:</p> <ul style="list-style-type: none"><li>● FILETRAN is the status monitor view for resources that support file transfers (used by the NetMaster for File Transfer product).</li><li>● IPRSC is the status monitor view for IP resources (used by the NetMaster for TCP/IP and NetSpy products).</li><li>● NCPVIEW is the status monitor view for NCPs (used by the NetMaster for SNA product).</li></ul>
List Name	Specify the name of the format.
Description	Provide a description for the format.
Service Procedure	Type <b>NONE</b> .
Entry Msg Position	Specify the column in which extended display starts. See the section, <i>Allowing for Extended Displays</i> , on page 8-9 for information about extended displays.
Entry Msg Length	<p>Specify the width of the extended display as follows:</p> <ul style="list-style-type: none"><li>● If you want the extended display to be of a specific width, specify the width in columns.</li><li>● If you want to allow the width to extend to the end of the screen, leave the field blank.</li><li>● If you do not want to show the extended display, specify 0.</li></ul>

Do *not* change the value of the other fields.

Figure 8-4 shows an example of the completed panel.

Figure 8-4. Status Monitor Display Format List Description

```

SOLVPROD----- CAS : List Description -----Page 1 of 4
Command ==>                                     Function=Add

Appl ID .....+ $RM
List Type ..... PRIVATE (PUBLIC or PRIVATE)
Userid ..... COLUMNS1 (Userid if PRIVATE)
List Name ..... ALLFLDS_
Description ..... All status fields monitor format
Title .....
Status ..... ACTIVE__ Group .....+
Service Procedure ..... NONE__ Data Source .....
Get All Entries? ..... YES Exit Name .....
Add Allowed? ..... YES Help Name .....
Default Mnemonic ..... B_ Select Mnemonic ..... S_
Entry Msg Position .... 33_ Entry Msg Length ....
Present Empty List? ... YES Auto Refresh Rate ...
Heading Sub Char ....

Comments .....

F1=Help      F2=Split    F3=File     F4=Save
              F8=Forward  F9=Swap
              F12=Cancel

```

- Step 3. Press F8 (Forward) three times to display the List Format panel. The panel provides a text editor window. To learn about the text editor, press F1 (Help).
- Step 4. By using the text editor, enter column headings and variables to specify the information to be displayed on the status monitor. See the next section, *Specifying the Status Monitor Display Format*, for information about how to specify the display format.
- Step 5. After you have completed your format specification, press F3 (File) to create the format.

## Specifying the Status Monitor Display Format

Specify the status monitor display format on the List Format panel.

The following list gives the widths for the five different column displays:

1-column display	75 characters
2-column display	35 characters
3-column display	21 characters
4-column display	15 characters
5-column display	11 characters

Each column contains an extra five positions at the beginning to allow for the command input field. For example, for a 1-column display, the width of the format is 75 characters, representing positions 6 through 80 of the status monitor.

For each type of information you want to display on the status monitor, you need to specify two items: a static heading and a variable that contains the required information.

Figure 8-5 shows the specification of the format supplied for the default 1-column display.

*Figure 8-5. Supplied 1-column Status Monitor Display Format*

```

SOLVPROD----- CAS : List Format -----Page 4 of 4
Command ==>                                     Function=Browse Scroll ==> CSR

Appl ID ... $RM      Type.Userid ... PRIVATE.COLUMNS1      Name ... DEFAULT

LINE ---+-----10-----20-----30-----40-----50-----60-----70-----+
**** ***** TOP OF DATA *****
0001 System Class Resource Desired Actual Mode Logical Ov
0002 &SYSNAME &CLNO &NAME &DSTST &PHYST &CURMODE &NRMST &O
**** ***** BOTTOM OF DATA *****
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

Heading Line ————

Entry Line ————

Specify the format as follows:

- Step 1. Specify the headings. A heading can be up to 10 lines. These lines are known as heading lines.
- Step 2. Specify the corresponding variables beneath the heading lines. Specify the variables in a single line, known as an entry line.

If the name of a variable is longer than the data to be displayed, create a shorter alias for the name.

The following sections provide detailed information that help you perform these steps.

## Specifying Headings

A heading describes the information being displayed under it. Type the headings as you would like them to appear on the status monitor. A heading can contain up to 10 lines of text.

## Using Variables

The variable contains the information you want to display. You can use any of the status variables. (See the *Automation Services Common User Guide* for a list of these variables.)

In addition, you can use the following special variables:

<b>Variable</b>	<b>Description</b>
&ZRMSTNRMCLDS	Contains the: <ul style="list-style-type: none"><li>• Subclass name for USRCLS class resources</li><li>• SNA group type for SNAGRP class resources</li><li>• Class name for services and other resources</li></ul> See also the &ZRMSTCLSNAME variable described in the <i>Automation Services Common User Guide</i> .
&ZRMSTOVRFLAG	Contains the flag that identifies the status overrides applied on a service or resource.
&ZRMSTXNAME	Contains the resource name. If the resource has an accompanying extended display, the name is prefixed with a plus sign (+).  If the display format hides extended displays, this variable enables you to identify the existence of an extended display, which you can view by entering S beside the resource.

For a full list of applicable variables, see help.

## Creating Shorter Aliases for Variable Names

The name of a variable can sometimes be longer than the data to be displayed. You can enter a shorter name and then make that shorter name an alias of the actual name.

Create aliases to variable names as follows:

- Step 1. From the List Format panel, press F5 (Fields).
- Step 2. On the displayed List Entry Line Fields panel, the Entry Line Field column contains the variable name you specified in the display format. Type the corresponding real variable name under the Real Field heading. Figure 8-6 shows the aliases used in the default 1-column display format.

*Figure 8-6. Variable Name Aliases Used in the Supplied 1-column Status Monitor Display Format*

```
SOLVPROD----- CAS : List Entry Line Fields -----
Command ==>                                     Function=Update Scroll ==> CSR

Appl ID ... $RM      Type.Userid ... PRIVATE.COLUMNS1      Name ... DEFAULT

Entry Line Field  Real Field
SYSNAME          ZRMSTSYSNAME
CLNO             ZRMSTNRMCLDS
NAME            ZRMSTNAME
DSTST           ZRMSTDSTACUR
PHYST           ZRMSTPHYSTAT
CURMODE         ZRMSTMODECUR
NRMST           ZRMSTNRMSTAT
OV              ZRMSTOVRFLAG
**END**

F1=Help      F2=Split    F3=File      F4=Save      F5=Format
F7=Backward  F8=Forward  F9=Swap      F12=Cancel
```

- Step 3. After you have created the aliases, you can perform one of the following actions:

<b>If you want to ...</b>	<b>Press ...</b>
Save the format and exit the format definition panels	F3 (File).
Save the format and remain on the List Entry Line Fields panel	F4 (Save).
Return to the List Format panel	F5 (Format).

## Allowing for Extended Displays

When extended displays are active, they overwrite the last portion of a formatted display screen. The starting position and the width of an extended display is determined by the values in the Entry Msg Position and the Entry Msg Length fields on the List Description panel (page 1 of the format definition panels).

With a multiscreen display format, extended displays affect the first screen only.

## Specifying a Multiscreen Display Format

You can create a multiscreen status monitor display. A multiscreen display can have up to 10 screens, enabling you to display more information on the status monitor. The screens can be accessed by pressing the F11 (Right) or F10 (Left) function keys from the status monitor.

Figure 8-7 shows an example that contains two screens. Lines 0001 and 0002 define the first screen, and lines 0003 and 0004 define the second screen.

*Figure 8-7. Multiscreen Status Monitor Display Format*

```
SOLVPROD----- CAS : List Format -----Page 4 of 4
Command ==>                                     Function=Browse Scroll ==> CSR

Appl ID ... $RM      Type.Userid ... PRIVATE.COLUMNS1      Name ... DSPFMT2P

LINE  ----+----10-----20-----30-----40-----50-----60-----70---
**** ***** TOP OF DATA *****
0001 System   Class   Resource   Desired  Actual   Mode     Logical  Ov
0002 &SYSNAME &CLNO   &NAME     &DSTST  &PHYST   &CURMODE &NRMST  &O
0003 Resource          Current Mode Reason
0004 &NAME              &ZRMSTMODECU0
**** ***** BOTTOM OF DATA *****
```

---

## What Is an Alert Monitor Display Format?

An alert monitor display format determines what information is displayed for the alerts on the alert monitor.

A default format is supplied, but you can define your own formats.

---

## Defining Alert Monitor Display Formats

Define alert monitor display formats from the List Definition List panel. To access the list, enter the `/ALADMIN.L` shortcut.

The alert monitor display formats are maintained as Common Application Services (CAS) lists. However, you *cannot* use the CAS application program interface to display the information specified by the display formats. (If you want to learn more about CAS lists, see the *Management Services Managed Object Development Services Programming and Administration Guide*.)

## Creating an Alert Monitor Display Format

Create an alert monitor display format from the List Definition List panel as follows:

- Step 1. Enter C beside the DEFAULT display format definition to display a copy that you can modify.
- Step 2. Type a new value in the List Name field to identify the new definition, and update the Description and Title fields.

Do *not* change the value of the other fields.

Figure 8-8 shows an example of the completed panel.

Figure 8-8. Alert Monitor Display Format List Description

```

SOLVPROD----- CAS : List Description -----Page 1 of 4
Command ==>                                     Function=Add

Appl ID .....+ $AM
List Type ..... PUBLIC_ (PUBLIC or PRIVATE)
Userid ..... (Userid if PRIVATE)
List Name ..... DEFAULT_
Description ..... Alert Monitor List Format_____
Title ..... Alert Monitor Default list Format_____
Status ..... ACTIVE_      Group .....+ _____
Service Procedure ..... $NDLH03Z      Data Source ..... $ALERTH_____
Get All Entries? ..... NO_      Exit Name ..... $AMSEALX_____
Add Allowed? ..... NO_      Help Name ..... _____
Default Mnemonic ..... DMP      Select Mnemonic ..... DMP_____
Entry Msg Position .... 2_      Entry Msg Length .... _____
Present Empty List? ... NO_      Auto Refresh Rate ... _____
                                   Heading Sub Char .... _____

Comments ..... _____
                                   _____
                                   _____

F1=Help      F2=Split      F3=File      F4=Save
              F8=Forward      F9=Swap
                                                    F12=Cancel
  
```

- Step 3. Press F8 (Forward) three times to display the List Format panel. The panel provides a text editor window. To learn about the text editor, press F1 (Help).
- Step 4. By using the text editor, enter column headings and variables to specify the information to be displayed on the alert monitor. See the next section, *Specifying the Alert Monitor Display Format*, for information about how to specify the display format.
- Step 5. After you have completed your format specification, press F3 (File) to create the format.

## Specifying the Alert Monitor Display Format

Specify the alert monitor display format on the List Format panel.

For each type of information you want to display on the alert monitor, you need to specify two items: a static heading and a variable that contains the required information.

You can create a multiscreen alert monitor display with up to 10 screens, enabling you to display more information on the monitor. The screens can be accessed by pressing the F11 (Right) or F10 (Left) function keys from the monitor.

### *Using Alert Variables*

The variable contains the information you want to display. See the Help for the list variables that contain alert information.

The name of a variable can sometimes be longer than the data to be displayed. You can enter a shorter name and then make that shorter name an alias of the actual name. To create aliases, press F5 (Fields) from the List Format panel.

---

## Maintaining Prompt Lists

Users of Automation Services can enter a question mark (?) in any prompted field to display a *prompt list* (that is, a list of valid values for the field). An authorized user can update these prompt lists if necessary. For example, you can update the Started Task Type prompt list to match the mix of started task types in your organization.

**This chapter contains the following topics:**

- Editing a Prompt List
- Maintaining List Entries

---

## Editing a Prompt List

To access the field prompt lists from the primary menu, enter the **/ASADMIN.P** shortcut. The Field Prompt List panel lists descriptions of all prompted fields, in alphabetical order.

### Adding a Value to a List

You can add an entry that will be used generically by a specific resource.

The following example explains how to add a new user-defined resource subclass to the User Classes prompt list. Sample panels are shown in Figure 9-1 on page 9-3.

To add a value to a list after accessing the Field Prompt List panel, complete these steps:

- Step 1. Type **L User Classes** at the command **====>** prompt to locate the User Classes field prompt.
- Step 2. Type **S** or **V** (FieldValues) beside User Classes, and press **ENTER** to display the Field Prompt Entry List panel which displays the existing values in the list.
- Step 3. Press **F4** (Add) to add a value, or enter **C** (Copy) beside an existing value to copy and modify it.

The Field Prompt Entry Definition screen is displayed.

- Step 4. Enter or modify the fields as required. Use variables, a string of less-than characters (<), or a string of greater-than characters (>) to define resource-specific details such as the resource name. You can also use the underline character () to define variables. (See the following sections for additional information.)
- Step 5. Use **F3** (File) to file the new value.

Figure 9-1. Adding a Value to a Prompt List

```

SOLVPROD----- Automation Services : Field Prompt List -----
Command ==>>>                                         Scroll ==>> PAGE

                                         S/V=FieldValues B=Browse U=Update D=Delete

Field Prompt Description
User Class (LOGICAL) - Status Monitor Message
v User Classes
**END**

SOLVPROD----- Automation Services : Field Prompt Entry List -----
Command ==>>>                                         Scroll ==>> PAGE

                                         S/B=Browse U=Update D=Delete C=Copy

-----
User Classes
-----
c L      LOGICAL   Logical Device
**END**

SOLVPROD--- Automation Services : Field Prompt Entry Definition ---$PROMPT-0000
Command ==>>>                                         Function=ADD

. Prompt Definition -----
|
| Description ... User Classes
|
|-----

. Prompt Entry Definition -----
|
| Abbrev ..... L_____
| FullValue .... LOGICAL_
| Description .. Logical Device_____
|
|-----

```

### Using Variables

Variables in a prompt list entry are substituted by their values when you select the entry for the field. You can disable variable substitution if you want the variable to appear in the field, *not* the value of the variable. Variable substitution can be disabled for variables with the formats &ZRM\* and &ZMSG\*.

To disable variable substitution, replace the ampersand (&) in front of the variable name with the underline character (\_). For example, if you specify \$\_ZRMDBNAME in a prompt list entry and select the entry for the field, &\$\_ZRMDBNAME is displayed in the field.

Refer to the *Automation Services Common User Guide* for the list of variables and the information they supply.

### Using Less-than Signs (<) to Represent a Left-justified, Fixed-length Variable Field Value

Some messages contain left-justified, fixed-length fields for resource names. If the name is not the right length, the name is left justified. To handle left justified fixed length fields, use less-than signs (<). Each less-than sign represents one character. (You cannot use variables because variables do not provide padding.) For example, <<<<< represents a 5-character field with left justification.

## Using Greater-than Signs (>) to Represent a Right-justified, Fixed-length Variable Field Value

Some messages contain right-justified, fixed-length fields for resource names. If the name is not the right length, the name is right justified. To handle right justified fixed length fields, use greater-than characters (>). Each greater-than sign represents one character. (You cannot use variables because variables do not provide padding.) For example, >>>>> represents a 5-character field with right justification.

## Maintaining Prompt List Definitions

When you apply the **U** (Update) action to an item on the field prompt list, the panel shown in Figure 9-2 is displayed.

Figure 9-2. Defining a Field Prompt

```
SOLVPROD----- Automation Services : Field Prompt Definition ---$PROMPT-0000
Command ==>>                                         Function=UPDATE

. Prompt Definition -----
|
| Name ..... $RM17TYPE
| Description ... User Classes
| Exclude Display
|-----

. Prompt Entry Specification -----
|
| Max Abbreviation Length ... 8   (3 - 8 or blank if none)
| Max Value Length ..... 8   (3 - 52)
| Max Description Length .... 38  (3 - 38 or blank if none)
|-----
```

The panel contains two windows. The Prompt Definition window identifies the prompt list, and the Prompt Entry Specification specifies the format of the values in the list. All the fields are modifiable.

### **Name**

Identifies the field.

### **Description**

Describes the prompt list. This description is displayed on the Field Prompt List panel to identify the prompt list. The description is also displayed as the heading on the Field Prompt Entry List panel, which is displayed when you enter a question mark (?) in a prompted field.

### **Exclude Display**

Specifies whether this definition is displayed on the Field Prompt List panel.

**Max Abbreviation Length**

Specifies the length of the abbreviated value of an entry in the prompt list.

Valid values are 3 through 8.

If you do not want to allow abbreviated values, leave the field blank.

**Max Value Length**

Specifies the length of the full value of an entry in the prompt list.

Valid values are 3 through 52.

**Max Description Length**

Specifies the length of the description of an entry in the prompt list.

Valid values are 3 through 38.

If you do not want to allow descriptions, leave the field blank.

**Note**

Ensure that the sum of the values in the Length fields results in a string that fits into a single line on the display screens. The string starts at column 7 on the display screens.

---

## Maintaining List Entries

Users with the appropriate authority can browse, update, and copy selection list entries.



# 10

---

## Defining Macros

This chapter describes how to define macros.

**This chapter contains the following topics:**

- What Is a Macro?
- Writing NCL Procedures to Be Used as Macros
- Registering and Maintaining the Macros

---

## What Is a Macro?

A macro is an NCL procedure used to construct process steps. Automation Services provides sample macros for your use, but you can also write your own NCL procedures and use them as macros. Your NCL procedure must be manually distributed to linked regions. To enable Automation Services to recognize an NCL procedure as a macro, you must register the macro.

Refer to the *Automation Services Common User Guide* for information on how to use a macro and for the list of sample macros.

---

## Writing NCL Procedures to Be Used as Macros

A typical NCL procedure that is to be used as a macro must have the following features:

- It must provide a brief description of the purpose of the procedure.
- It must have display panels that allow users to browse or change certain parameters used by the procedure. These are displayed when a user applies the **P** (Parameters) action.
- It must perform the required function when a process executes the macro.
- It must return a code that a user can test to determine what the processing requirements are for later process steps.

Automation Services includes two macro template procedures in the *dsnpref.ASVVV.ASTEXEC* data set—`$RMMC00S` (which executes macro actions) and `$RMMC00D` (which handles parameter definitions)—for you to copy to the `TESTEXEC` data set and customize to suit your requirements.

## Customizing the \$RMMC00S Macro Template

Firstly, you need to set the value of `#RMMACRO` (that is, the macro name). Next, the subroutines introduced by the following labels should be customized:

- `SHORTDESC` (optional)
- `LONGDESC` (optional)
- `PARMHELP` (optional)
- `EXECMACRO` (*mandatory*)

Further customization can be performed, but these are the recommended minimum changes to the `$RMMC00S` template.

## Customizing the \$RMMC00D Macro Template

The subroutines introduced by the following labels must be customized, if you want parameter validation to work correctly:

- PINIT
- PSET
- PSAVE
- VMANDATORY

Further customization can be performed, but these are the recommended minimum changes to the \$RMMC00D template.

---

## Registering and Maintaining the Macros

To enable an NCL procedure to be used as a macro, register the macro by creating a macro definition for the procedure. Use the Macro Definition panel to define the macro.

## Accessing the Macro Definitions

Enter the /ASADMIN.M shortcut on the primary menu to display the Macro List panel. A sample list of macro definitions is shown in Figure 10-1.

*Figure 10-1. A Macro List*

```
SOLVPROD----- Automation Services : Macro List -----
Command ==>                                         Scroll ==> PAGE

                                     S/B=Browse U=Update C=Copy D=Delete
Macro      NCL Proc  Description
CHAIN      $RMMC06S  Chain to a Process
COMMAND    $RMMC04S  Execute a SOLVE Command
EXECNCL    $RMMC09S  Execute an NCL Procedure
EXTRACT    $RMMC17S  Extract a Portion of a Data String
PARSE      $RMMC15S  Segment a String to Create Vars
REPLY      $RMMC03S  Reply to an Outstanding WTOR
RUNPRCSS   $RMMC08S  Execute a Process
SETRC      $RMMC01S  Set Return Code
SETSTATE   $RMMC18S  Set Resource Monitor Display for a Resource
SETVARS    $RMMC14S  Create variables for a Process
STARTNCL   $RMMC10S  Start an NCL Procedure
STOP       $RMMC05S  Set Return Code and End
SUBJOB     $RMMC11S  Submit JCL to Run a Job
SUBPRCSS   $RMMC07S  Submit a Process to a Background Region
SVAPI      $RMMC12S  Allow access to ServiceView API from a Process
SVCMD      $RMMC19S  Execute a ServiceView Command
SYSCMD     $RMMC02S  Execute a System Command
```

The panel displays the list of available macros. Authorized users can browse, add, update, copy, and delete macro definitions.

## The Macro Definition Panel

The Macro Definition panel records macro details and identifies the NCL procedure represented by the macro. Depending on the operation you are performing, some or all of the fields on this panel might be modifiable. (If you need help with the field definitions, press F1 (Help).)

Figure 10-2 shows a macro definition in the process of being updated.

*Figure 10-2. Updating a Macro Definition*

```
SOLVPROD----- Automation Services : Macro Definition -----$$MAC-0000
Command ==>                                                    Function=UPDATE

. Macro Definition -----
| Macro ..... CHAIN
| NCL Procedure ..... $RMMC06S
| Description ..... Chain to a Process
| (Description will be automatically set by the Macro)
|-----

Macro History

      Created                Last Updated
      Userid ... USER01      Userid ... USER07
      Date ..... WED 19-OCT-1994  Date ..... WED 20-SEP-1995
      Time ..... 13.10.55         Time ..... 15.33.47

F1=Help      F2=Split      F3=File      F4=Save
              F9=Swap
                                  F12=Cancel
```

## Adding and Maintaining Macro Definitions

Macro definitions are added by pressing F4 (Add) on the Macro List panel, then completing the Macro and NCL Procedure fields on the Macro Definition panel.

You can subsequently update a macro definition so that it points to a different NCL procedure. The NCL Procedure field is the only updateable field on the panel.

Macro definitions can also be browsed, copied, or deleted from the macro list.

---

## Defining Commands

This chapter describes how to define Automation Services commands.

**This chapter contains the following topics:**

- What Is an Automation Services Command?
- Registering and Maintaining Commands
- Defining Your Own Commands

---

## What Is an Automation Services Command?

A command is an NCL procedure that performs specific processing. Automation Services comes with a predefined set of commands.

Commands can be generic (that is, perform processing that is not related to a specific resource or service) or can apply to a class, to a type within a class, or to a specific resource or service. A more specific command has precedence over a less specific command.

You can define your own commands to perform site-specific processing by writing an NCL procedure and associating it with a command definition. Supplied command definitions can be modified to suit your requirements.

To execute commands programmatically from a batch process, exit procedure, console, or external application, use the \$RMCALL API procedure. (See Appendix G, *Using the Application Program Interface*, for details.)

## Registering and Maintaining Commands

Automation Services does not recognize commands unless they are registered by using the command definition facility.

Use this facility to register your own command definitions and to maintain command definitions supplied with Automation Services.

### Accessing Command Definitions

Enter the `/ASADMIN.C` shortcut on the primary menu to access the Command List panel (shown in Figure 11-1).

This panel lists all the commands that are currently defined, the type of resource or service to which each command applies, and a description of the command. Special classes include ALL, ALL-X, INC, and NONE. For information about these classes, press F1 (Help).

Figure 11-1. A Command List

```
SOLVPROD----- Automation Services : Command List -----
Command ==>>>                                     Scroll ==>> PAGE

                                     S/B=Browse U=Update C=Copy D=Delete
Command   Class  Type      Description
A         ALL-X   SNA       Activate a Resource
A         SNA     SNA       Activate an SNA Resource
ACKLNKFAIL NONE     NONE      Acknowledge a Link Failure
ADD       NONE     NONE      Add a Resource
AFF       ALL-X   NONE      Browse a Shared Resource's Affinity
ALF       NONE     NONE      Acknowledge a Link Failure
ASA       ALL-X   NONE      Set Actual State to Active
ASD       ALL-X   NONE      Set Actual State to Degraded
ASD       SNA     SNA       Set Resource state to DEGRADED
ASF       ALL-X   NONE      Set Actual State to Failed
ASF       SNA     SNA       Set Resource state to FAILED
ASI       ALL-X   NONE      Set Actual State to Inactive
ASU       ALL-X   NONE      Set Actual State to Unknown
ATD       SNA     SNA       NEWS Real-time Attentions display
ATL       SNA     SNA       Display NEWS Attentions List
B         ALL     NONE      Browse Resource Status and Modes
BLD       SNAGRP  SNA       Rebuild an SNA Group
```

Depending on your level of authority, you can either add, update, copy, and delete command definitions, or just browse these definitions.

## The Command Details Panel

The Command Details panel is used to specify:

- The name and description of the command
- Whether the command applies to a class, a type within a class, or a specific resource or service

### Note

For duplicate commands, the region applies the command according to their scope. For example, if you enter A beside an SNA resource, the A command with the class scope of SNA takes precedence over the A command with the class scope of ALL-X.

- The user ID under which the command executes
- Whether the command is executed on remote or local region
- The name of the NCL procedure that performs the processing associated with the command, and the names of any parameters passed to this procedure

Figure 11-2 shows an example of the Command Details panel in Browse mode. For more information about the panel, press F1 (Help).

*Figure 11-2. Browsing the Command Details Panel*

```
SOLVPROD----- Automation Services : Command Details -----$$CMD-0000
Command ==>                                         Function=BROWSE

. Command Identification -----
Name ... ASA           Description ... Set Actual State to Active
-----

. Command Scope -----
Class ...+ ALL        Type ...           Name ...
-----

. Execution Details -----
Command Userid ...+ *           Execute on Remote? ... YES (YES or NO)
NCL Procedure ... $RMCONS LCMD=ASA SYSNAME=&ZRMDBSYSNAME
                                VERSION=&ZRMDBVERSION
                                CLASS=&ZRMDBCLASS
                                NAME=&ZRMDBNAME
-----

F1=Help      F2=Split    F3=Exit     F4=Edit     F5=Classes
             F8=Forward  F9=Swap    F11=Panels
```

## The Command Prompting, Confirmation and Validation Panel

The Command Prompting, Confirmation and Validation panel is used to specify:

- Whether the user is prompted, and the type of prompting, if parameters required by the command are not supplied
- Whether confirmation is required before the command is executed and the name of the (optional) confirmation exit procedure that performs the confirmation
- The name of the (optional) validation exit procedure that performs validation on command parameters

Figure 11-3 shows an example of the Command Prompting, Confirmation and Validation panel in Browse mode. For information about the field definitions, press F1 (Help).

*Figure 11-3. Browsing the Command Prompting, Confirmation and Validation Panel*

```
----- Automation Services : Prompting, Confirmation and Validation -----
Command ==>                                     Function=BROWSE

. Prompting -----
|
| Type of Prompting Required ... RESOURCE (SYSTEM, RESOURCE or NONE)
|
| SYSTEM  prompts for System Name and Version if not provided
| RESOURCE prompts for System Name, Version, Class and Name if not provided
| NONE    defaults to the local System Name and Version if not provided
|-----
. Confirmation -----
|
| Does This Command Require Confirmation? ... YES (YES or NO)
| Confirmation Exit ...
|-----
. Validation -----
|
| Validation Exit .....
|-----

F1=Help      F2=Split      F3=Exit      F4=Edit
F7=Backward  F9=Swap          F11=Panels
```

## Adding and Maintaining Command Definitions

Command definitions are added by pressing F4 (Add) on the Command List panel, then completing the mandatory fields on the Command Details panel.

### Note

If you want to define a line command, the name can only be three characters long.

Command definitions can also be browsed, updated, copied, or deleted from the Command List panel.

---

## Defining Your Own Commands

You can define your own commands to perform specialized processing. You need to:

- Write an NCL procedure that performs the processing that you want your command to perform
- Register the command to Automation Services (described in the previous section)
- Manually distribute the procedure to linked regions

You can register different versions of a command with the same name. Automation Services executes the version that best fits the resource or service the command is applied to.

For example, you can register two versions of a command with the name Z, where version one applies to all resources, and version two applies to started tasks only. When the Z command is applied to a started task, Automation Services executes version two of the command.

## Coding Your NCL Procedure

Your NCL procedure should set RETCODE=0 to indicate successful completion—a non-zero return code indicates a processing failure. You can return a message from the procedure by setting the &SYSMSG variable.

For further information on NCL programming, see the *Network Control Language User Guide* and the *Network Control Language Reference* manuals.

## Commands That Execute on Remote Systems

If a command relates to a resource or service on a remote system, you can set the Execute on Remote? field to YES. The NCL procedure associated with the command will then execute on the remote system.

If the command requires a presentation space (that is, the command displays information to the user who issued it), set the Execute on Remote? field to NO. In this case, your NCL procedure must extract data from the remote system and present it on the local system. The name of the system where the data is sourced is stored in the variable ZRMCMDLINK.

## Variables Available to a Command NCL Procedure

The following variables can be used in an NCL procedure associated with a command, to pass data required by the command:

### **ZRMSYSNAME**

Contains the name of the system to which the command is to be applied.

### **ZRMVERSION**

Contains the version number of the system to which the command is to be applied.

### **ZRMCMDUSERID**

Contains the user ID of the person who issued the command.

### **ZRMCMDLINK**

Contains the name of a link (which is, in fact, the ACB name of the target region) that will route a command to a region. The ZRMCMDLINK variable contains the ACB name of the current region if the command is not to be routed to a different system.

### **ZRMCMDPARMS**

Contains any command parameters specified by the user. Parameters are in the format *parameter=value*, separated by spaces. This enables you to override default values.

**ZRMDB\***

Knowledge base variables have the prefix ZRMDB. If the command is to be applied to a resource or a service defined in the knowledge base, the command procedure needs access to relevant values stored in knowledge base variables. One or more of the following four variables is likely to be required:

**ZRMDBSYSNAME**

Contains the name of the system image where the resource is defined if the command refers to a specific resource or service.

**ZRMDBVERSION**

Contains the version number of the system image where the resource is defined if the command refers to a specific resource or service.

**ZRMDBCLASS**

Contains the class name of the resource or service to which the command is to be applied.

**ZRMDBNAME**

Contains the name of the resource or service to which the command is to be applied.

Further knowledge base variables are listed in the *Automation Services Common User Guide*.

**ZRMST\***

Status variables have the prefix ZRMST. If the command is to be applied to a defined resource or service, the command procedure needs access to relevant values stored in status table variables. See the *Automation Services Common User Guide* for a list of available variables.

**ZMSG\***

Message variables have the prefix ZMSG. If the command is to be executed as a result of the receipt of a particular message, the command procedure needs access to relevant values stored in message variables. See the *Automation Services Common User Guide* for a list of available variables.

# Part II

---

## Reference Material



# A

---

## Security Settings

This appendix specifies the recommended security settings for each of the supplied group IDs. These settings can be used if you want to create group IDs for your external security package. The settings are presented as they are in UAMS providing the panel, field, value, and structured field for each setting.

**Note**

The structured field information will only be useful to installations that have an external security package controlling user access with a partial or full security exit. See the *Management Services Administrator Guide* for detailed information about the structured fields.

## Security Settings for \$RMADMIN

\$RMADMIN is the group ID for administrators. The recommended security settings are given in Table A-1.

Table A-1. Security Settings for \$RMADMIN

Panel (Page Number)	Field	Value	Structured Field
User Authorities (2)	Authority Level	255	50
	Multiple Signon Authority	Y	19
	Split/Swap Authority	Y	2E
	APPC Access Key	ALL	60
	APPC Access Lock	ALL	61
User Access (3)	Network Management	Y (license dependent)	22
	Operations Management	Y	2B
	Operator Console Services	Y	20
	Management Services	Y	511
	UAMS Maintenance	Y	2A
	Broadcast Services	Y	21
	System Support Services	Y	23
OCS Details (6)	Monitor Status	Y	51
	Message Code	FF	58
	NPF Resource List Member	\$RMSXADM	5B
	Initial OCS Command	-\$RMCCOCS	54

(Sheet 1 of 2)

Table A-1. Security Settings for \$RMADMIN

Panel (Page Number)	Field	Value	Structured Field
Network Management Details (8)	If licensed, then Y should be specified for those features that are enabled		
AOM General Details (11)	AOM Message Receipt	Y	180
	Console Routing Codes	ALL	180B
	Message Level Screening	ALL	182
AOM MVS Details (12)	Console Authority	M	181
Print Services Manager Details (13)	Maintenance Access	2	501

*(Sheet 2 of 2)*

## Security Settings for \$RMOPER

\$RMOPER is the group ID for operators. The recommended security settings are given in Table A-2.

Table A-2. Security Settings for \$RMOPER

Panel (Page Number)	Field	Value	Structured Field
User Authorities (2)	Authority Level	200	50
	Multiple Signon Authority	Y	19
	Split/Swap Authority	Y	2E
	APPC Access Key	ALL	60
	APPC Access Lock	ALL	61

*(Sheet 1 of 2)*

Table A-2. Security Settings for \$RMOPER

Panel (Page Number)	Field	Value	Structured Field
User Access (3)	Network Management	Y (license dependent)	22
	Operations Management	Y	2B
	Operator Console Services	Y	20
	Management Services	Y	511
	Broadcast Services	Y	21
OCS Details (6)	Monitor Status	Y	51
	Message Code	FF	58
	NPF Resource List Member	\$RMSXOPR	5B
	Initial OCS Command	-\$RMCCOCS	54
Network Management Details (8)	If licensed, then Y should be specified for those features that are enabled		
AOM General Details (11)	AOM Message Receipt	Y	180
	Console Routing Codes	ALL	180B
	Message Level Screening	ALL	182
AOM MVS Details (12)	Console Authority	M	181

(Sheet 2 of 2)

## Security Settings for \$RMNOPER

\$RMNOPER is the group ID for network operators. The recommended security settings are given in Table A-3.

Table A-3. Security Settings for \$RMNOPER

Panel (Page Number)	Field	Value	Structured Field
User Authorities (2)	Authority Level	200	50
	Multiple Signon Authority	Y	19
	Split/Swap Authority	Y	2E
	APPC Access Key	ALL	60
	APPC Access Lock	ALL	61
User Access (3)	Network Management	Y	22
	Operator Console Services	Y	20
	Broadcast Services	Y	21
OCS Details (6)	Monitor Status	Y	51
	Message Code	FF	58
	NPF Resource List Member	\$RMSXNOPR	5B
	Initial OCS Command	-\$RMCCOCS	54
Network Management Details (8)	If licensed, then Y should be specified for those features that are enabled		

---

## Security Settings for \$RMMON

\$RMMON is the group ID for monitors. The recommended security settings are given in Table A-4.

*Table A-4. Security Settings for \$RMMON*

---

<b>Panel (Page Number)</b>	<b>Field</b>	<b>Value</b>	<b>Structured Field</b>
User Authorities (2)	Authority Level	000	50
	Multiple Signon Authority	Y	19
	APPC Access Key	ALL	60
	APPC Access Lock	ALL	61
User Access (3)	Network Management	Y (license dependent)	22
	Operations Management	Y	2B
AOM General Details (11)	AOM Message Receipt	Y	180
	Console Routing Codes	ALL	180B
	Message Level Screening	ALL	182
AOM MVS Details (12)	Console Authority	I	181

---

## Security Settings for \$RMBUSER

\$RMBUSER is the group ID for background users. The recommended security settings are given in Table A-5.

Table A-5. Security Settings for \$RMBUSER

Panel (Page Number)	Field	Value	Structured Field
User Authorities (2)	Authority Level	255	50
	APPC Access Key	ALL	60
	APPC Access Lock	ALL	61
User Access (3)	Network Management	Y (license dependent)	22
	Operations Management	Y	2B
	Operator Console Services	Y	20
	Management Services	Y	511
	Broadcast Services	Y	21
	Object Services Support	Y	605
	System Support Services	Y	23
OCS Details (6)	Monitor Status	Y	51
	NPF Resource List Member	\$RMSXADM	5B
Network Management Details (8)	If licensed, then Y should be specified for those features that are enabled		
AOM MVS Details (12)	Console Authority	M	181



# B

---

## Parameter Groups

System parameters are grouped, by category, in logical parameter groups, to simplify the initialisation and customization of a region. This appendix lists the various categories of system parameter groups you can set. The setup methods are described in Chapter 2, *Initializing, Implementing, and Customizing a Region*.

---

## List of Parameter Groups

The ICS parameter groups are listed in Table B-1. Groups that are specific to a particular product are identified in parentheses. See help for other groups not covered in this table.

Table B-1. ICS Parameter Groups

---

Category	Parameter Group ID	Description
Files	AUTOFILES	Automation Files Specification—identifies the: <ul style="list-style-type: none"><li>• Knowledge base</li><li>• Staging file</li></ul> The following parameters in this group do not apply on VM systems: <ul style="list-style-type: none"><li>• Internal reader</li><li>• Default JCL library for jobs</li></ul>
	LOGFILES	Log File Specifications—specifies the activity log requirements.
	MODSFILES	MODS Files Specifications—identifies the required MODS files.
	NETINFODB	NetInfo Database Specification—identifies the NETINFO file.
	PANELLIBS	Panel Libraries Specifications—specifies the panel library paths.
	PSMSPOOL	PSM Spool File Specification—identifies the PSM spool file.

---

(Sheet 1 of 5)

Table B-1. ICS Parameter Groups

Category	Parameter Group ID	Description
Interfaces	CICSCNTL (CICS)	CICS Connections—specifies the PPI connections to CICS regions.
	EXTAPPLS	External Applications Access—specifies the: <ul style="list-style-type: none"> <li>• External applications access requirements</li> <li>• LU 1 logon mode for the MTO command</li> </ul>
	SOCKETS	TCP/IP Sockets Interface—specifies the TCP/IP software interface requirements.
	SSI (not applicable on VM)	SOLVE Sub-System Interface—specifies the: <ul style="list-style-type: none"> <li>• Subsystem interface requirements</li> <li>• Non-VTAM terminal requirements (OS/390 systems only)</li> </ul>
	TELNETSRVR	Telnet Server Controls—specifies the Telnet access requirements.
	UNICENTER	Unicenter Agent Specification—configures the agent that interfaces with the Unicenter TNG Framework for OS/390 software.
	WSPEER	Workstation Peer Specification—specifies the requirements for connecting to SOLVE:Operations for OpenView and IT/Operations processes.

(Sheet 2 of 5)

Table B-1. ICS Parameter Groups

Category	Parameter Group ID	Description
Names	AUTOIDS	Automation Identifiers—specifies the: <ul style="list-style-type: none"> <li>• System image to be loaded on startup</li> <li>• Default desired state</li> <li>• Ruleset actions in MANUAL global operation mode</li> <li>• Checkpoint requirements</li> <li>• Startup WTOR requirement</li> </ul>
	OPSYSIDS	Operating System Identifiers—identifies the: <ul style="list-style-type: none"> <li>• Template system image</li> <li>• System</li> <li>• Logon mode for APPC sessions</li> </ul> <p>The following parameters in this group do not apply on VM systems:</p> <ul style="list-style-type: none"> <li>• JES and its command character</li> <li>• AOM subsystem interface and its command character</li> </ul> <p>The following parameters in this group apply to VM systems only: programmable operator IDs.</p>
	SYSTEMID	System Identifications—specifies the: <ul style="list-style-type: none"> <li>• Region ID</li> <li>• Logon panel title</li> <li>• Message monitor title</li> </ul>
Security	CMDREPLS	Command Replacements—specifies the replaced MS commands.
	LOGONUSRDATA	Should Region Accept User Data?—specifies whether to accept user data when a user logs on to the region.
	SEC SHIPPING	Ship UAMS Maintenance—specifies the UAMS synchronization requirements.

(Sheet 3 of 5)

Table B-1. ICS Parameter Groups

Category	Parameter Group ID	Description
Tuning	ABENDCMD (not available on VM)	Command Issued if System ABENDs—specifies the system command to issue when the region ends abnormally.
	AOMQUEUES	AOM System Queue Limits—specifies the: <ul style="list-style-type: none"> <li>• AOM SSI storage requirements for message traffic</li> <li>• Threshold that prevents message bursts from overloading the Automation Services region</li> </ul>
	AUTOTABLES	Automation Table Controls—specifies the: <ul style="list-style-type: none"> <li>• Default size of a transient log</li> <li>• Size of the vartable for the active system image</li> <li>• Command cache requirements</li> <li>• Size of the vartable for message learning</li> </ul>
	CONSOLES	Console Specifications—specifies the: <ul style="list-style-type: none"> <li>• Console requirements</li> <li>• System command processing requirements</li> </ul>
	LSRPOOL	VSAM LSR Pool Specifications—specifies the local shared resource (LSR) pools of buffers.
	MULTISYS	Multi-System Options—specifies whether multisystem operations is supported by one or more of the following: EPS, TCP/IP, and VTAM.
	NCLCONTROLS	NCL Control Parameters—specifies the NCL processing environment.
	NDBLIMITS	NDB Processing Limits—specifies the NDB scanning limits.
	NONSWAP (not available on VM)	Should Region Run Non-swappable?—specifies whether the region should ever be swapped out of memory.
	SMFDATA (not available on VM)	SMF Recording Details—specifies the SMF (or SMS) recording details.
	TIMINGS	Automation Timing Controls—specifies the: <ul style="list-style-type: none"> <li>• Status display timing requirements</li> <li>• Link processing timing requirements</li> </ul>

(Sheet 4 of 5)

Table B-1. ICS Parameter Groups

Category	Parameter Group ID	Description
Useability	CCONSOLIDATN	Console Consolidation Options—specifies: <ul style="list-style-type: none"> <li>• Whether console consolidation is enabled</li> <li>• Message profile restrictions</li> <li>• When to deliver messages to remote regions, immediately or after an EventView ruleset is loaded</li> </ul>
	DISPLAYS	Presentation Controls—specifies certain characteristics of the graphical monitor and dynamic displays.
	EQUATES	System Equates—specifies equated strings.
	EVENTSIM	Event Simulation Options—specifies whether the event simulator is enabled.
	PMENUCONTROL	Primary Menu Control Options—specifies how the primary menu is displayed.
	TERMDEFS (VOS3)	Terminal Definitions—specifies the type and attributes of the terminals that may be used as Automation Services consoles.

*(Sheet 5 of 5)*

# C

---

## Consoles

For operators to be able to issue system commands from an Automation Services region, it is essential that you define suitable consoles to the system before starting the region. An Automation Services region can use JES or extended multiple console support (MCS) consoles in an OS/390 environment, OP1 or OP2 consoles in an MSP environment, and pseudo consoles in a VOS3 environment.

This appendix describes these consoles as used by an Automation Services region.

---

## JES, OP1, OP2, and Pseudo Consoles

JES, OP1, OP2, and pseudo consoles are virtual consoles. They can be acquired by any authorized program for use in issuing system and subsystem commands.

---

## Extended MCS Consoles

In the OS/390 environment, Automation Services can use extended MCS virtual consoles.

The advantages in using extended MCS consoles are:

- There is no theoretical limit to the number of extended MCS consoles in an OS/390 configuration. JES consoles are limited to 99 across a sysplex.
- You can have a MASTER authority level.
- In a sysplex configuration, extended MCS consoles are more flexible.

Automation Services uses extended MCS consoles as follows:

- The 8-character name of consoles are constructed by using the convention specified in the CONSOLES parameter group. The extended MCS console prefix is further prefixed with a Z and padded to five characters with Zs. The last three characters are a decimal number from 001 through 255. The region can use up to 254 extended MCS consoles.
- Console authority of MASTER can be set and honored. (SYSCMD CON=MASTER is ignored.)

## Migration IDs

It is possible to assign a unique migration ID to an extended MCS console. The ID is required in order to issue commands to some applications. Across a sysplex, there is a limit of 150 migration IDs.

When a region issues commands internally by using an extended MCS console, it decides whether a migration ID is required through its migration ID determination exit. The exit requests an ID for all MODIFY and STOP system commands, and for any unrecognized commands.

When users issue commands in the region by using the SYSCMD command, they can choose whether a migration ID is required by using the MIGID operand of the SYSCMD command. If the MIGID operand is not specified, the setting specified in the CONSOLES parameter group is used. The default setting is YES, specifying that IDs be used. (For the syntax of the SYSCMD command, see the Help or the *Command Reference*.)

The limit of 150 migration IDs might be a problem in a large sysplex where many applications are using extended MCS consoles. If a region cannot acquire a console when requested, the requesting SYSCMD command will fail. Use the following suggestions to help you correct the problem:

- In the CONSOLES parameter group, change the value in the Acquire with Migration ID (default) field to NO.
- Consider writing your own migration ID determination exit to further restrict the use of IDs by internally issued commands.

A sample exit, NMMIGIDX, is supplied with the product. For more information, see the *Management Services Administrator Guide*.

---

## Console Management

A region uses the CONSOLES parameter group to specify its console requirements.

Automation Services uses consoles as follows:

- During region initialization, the CONSOLES parameter group sets a limit on the number of consoles available to authorized users in the region.
- Consoles are acquired and released as necessary (see the next section, *Console Pool Management*).

## Console Pool Management

The CONSOLES parameter group contains the following two values that govern how the region manages its pool of consoles:

- Maximum number of consoles that can be acquired concurrently by the region
- Maximum number of free consoles that should be retained in a pool for the region when the region finishes using them

## Console Acquisition

When a user issues the SYSCMD command, the region tries to acquire a console as follows:

- It uses a console previously assigned to the user if the console has not timed out and if the console attributes match the new request.
- It searches the pool of consoles for a free console. If one is found, it is temporarily assigned to the environment that is issuing the command.
- If there are no free consoles and if the number of consoles currently acquired is less than the specified maximum, it tries to acquire one from the system.
- If the maximum number of consoles are already acquired, the acquisition fails. For commands issued internally, the region will retry the acquisition as specified by the CONSOLES parameter group.

## Console Release

When a console is not used any more, it is put in the pool. When the number of free consoles exceeds the value specified in the CONSOLES parameter group, the extra consoles are released back to the system.

---

## Specifying the Number of Consoles

The normal CPU consumption profile of a region takes the shape of a valley between two peaks. The peaks occur during region startup and shutdown, and the valley occurs during normal operation. You should allocate the appropriate number of consoles to handle the consumption peaks. For example, Figure C-1 shows the settings in the CONSOLES parameter group for a production region.

### Note

Because more extended MCS consoles are available than JES consoles, use extended MCS consoles in preference to JES consoles when it is feasible.

*Figure C-1. Console Settings for a Production Region*

```
-----  
.- CONSOLES - Console Specifications -----  
|  
| Type of Consoles to Acquire .....+ EXTMCS  
| Max Consoles to Acquire ..... 40_  
| Max Consoles to Retain ..... 30_  
|  
-----
```

Use the SHOW CONSOLES command to review console usage by the region.

## Considerations

If a region requires a large number of consoles (for a large number of defined resources), then the larger the number of consoles retained, the more efficient the region becomes. However, retained consoles are not available to other applications.

Consider the following when specifying the Max Consoles to Retain value in the CONSOLES parameter group:

- The number of MSP OP1 and OP2 consoles are limited to 10 per system.
- The number of OS/390 JES consoles are limited to 99 across a sysplex.
- It might be necessary to limit the number of consoles to retain if a significant number of extended MCS consoles require migration IDs.



# D

---

## Non-VTAM Terminal Support

There are times when you want to be able to use terminals to communicate with regions that use Automation Services while VTAM is not available. For example, during system IPL, you might want to log on and run automation in full-screen mode without waiting for VTAM to become active.

This appendix provides an overview of the non-VTAM terminal support feature that provides this functionality and how to implement it. A region can provide non-VTAM support in the following ways:

- By using a local terminal as a non-VTAM terminal
- By using Telnet

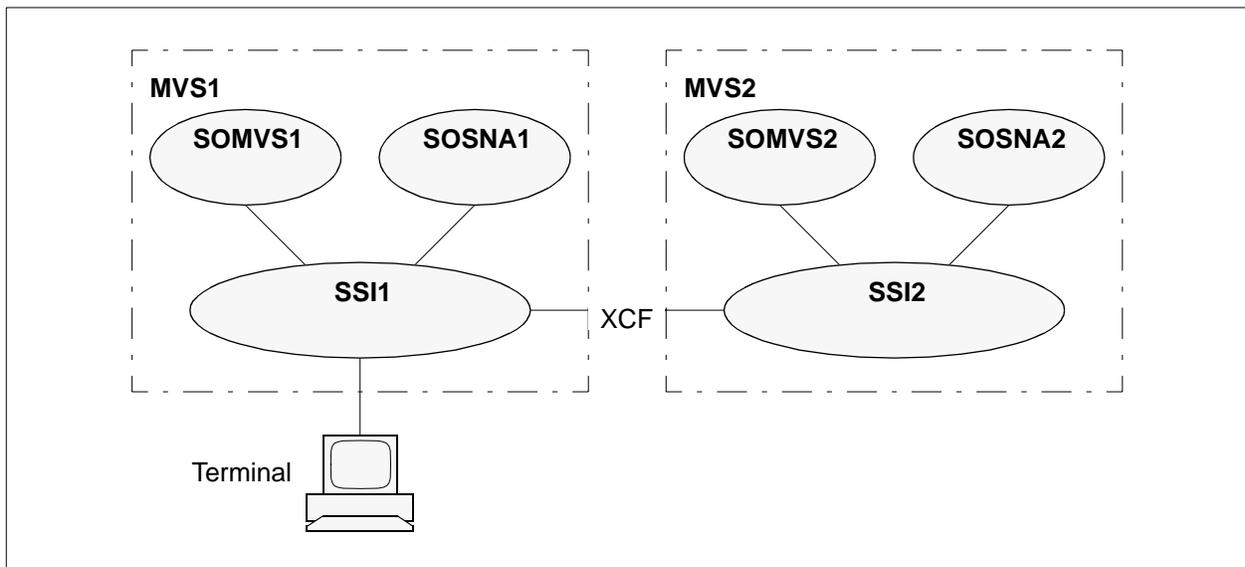
## Implementing Non-VTAM Terminal Support by Using a Local Terminal

Non-VTAM terminal support is implemented by using the SSI (subsystem interface) access method. This method supports communication between a local terminal attached to a program called NMSSI and up to 16 regions.

### Sysplex Support

In a sysplex environment, you can use the cross-system coupling facility (XCF) to enable a local terminal to access a region on another system. Figure D-1 shows an example.

Figure D-1. Non-VTAM Terminal Support in a Sysplex Environment



To register the SSI region to the XCF component, add the XCF=YES parameter in the *dsnpref.SIvvv.SSIPARM(SSIPARMS)* member.

## Enabling Terminal Support in SSI

An SSI startup parameter, `TERMINALS={YES|NO}`, is used to specify whether the NMSSI program is to provide support for non-VTAM terminals.

If `TERMINALS=YES` is specified, communications for terminals are initialized when the NMSSI program is initialized.

The `TERMINALS` parameter is specified in the `SSIPARMS` member. If you want to implement non-VTAM terminal support you need to set this parameter to `YES`.

### Note

If you are running multiple SSI regions on a single system, only one of them may be set up for non-VTAM terminals.

You need to review other startup parameters for the SSI (in particular the `SSID` parameter). See the *Management Services Administrator Guide* for a full discussion of SSI startup parameters and other SSI implementation details.

## Specifying Accessible Regions

To specify the regions that can be accessed by the terminals attached to an SSI region, add `TERMACCESS` parameters to the `SSIPARMS` member.

You can specify up to 16 `TERMACCESS` parameter statements.

### Enabling Automatic Logon to a Region

The automatic logon parameter for a non-VTAM terminal is set in the SSI parameter group.

When automatic logon is enabled for a terminal, the terminal is automatically logged on to the region specified in the first `TERMACCESS` parameter statement only.

## TERMACCESS Parameter

A TERMACCESS parameter specifies the region that a non-VTAM terminal can access and the function key that is used to access the region.

```
TERMACCESS=(PFnn,solve-region-id,description)
```

### **PFnn**

Specifies the function key that is used to access the *solve-region-id* region.

*nn* must be in the ranges 5 through 12 and 17 through 24.

### ***solve-region-id***

Specifies the ID of a region that a terminal can access. If you registered the SSI regions with the XCF component in a sysplex environment, the region does not need to be on the same system.

*solve-region-id* is specified in the SYSTEMID parameter group. The default is the value specified for the PRI=JCL parameter in the RUNSYSIN member.

### ***description***

Provides a short description to identify the region that a function key accesses. This description is displayed on a terminal function key menu.

*description* is 1 through 20 characters long. It does not support quoted strings. To represent a blank character, use an underscore.

### *Example*

The following example enables you to press PF11 to access the SOLVMVS1 region from a non-VTAM terminal.

```
TERMACCESS=(PF11,SOLVMVS1,SOLVE:OPS_MVS_1)
```

## Attaching and Detaching Terminals

The ATTACH and DETACH commands can be used to attach terminals to and detach terminals from NMSSI.

When you use these commands, you need to specify the device address of the terminal you want to attach or detach. NMSSI uses this address to dynamically allocate and deallocate the device.

### Note

These commands are available from a region only if its AM operand includes SSI as one of the access methods.

See the section, *Commands*, on page D-10 in this appendix for further details.

## Defining Terminal Names

Non-VTAM terminal support allows both 3-character and 4-character terminal addresses.

NMSSI assigns an eight-character symbolic name to a terminal that consists of a 1 to 5-byte prefix followed by the terminal device address.

If a terminal has a 4-character address and the first character is not zero (for example, 1FFF), you must limit the length of the prefix to four characters.

The symbolic name prefix defaults to \$LOCL—you can, however, specify a different prefix using the TPREFIX operand in NMSSI startup parameters.

You can also specify a terminal name through the NAME operand of the ATTACH command. This name overrides the name specified in the TPREFIX operand for the terminal being attached.

## Activating the SSI

You need to activate the SSI before you can use the non-VTAM terminal support facility.

See the chapter about the subsystem interface (SSI) in the *Management Services Administrator Guide* for details.

## Displaying Attached Terminals

The SHOW SSITERMS command displays terminals attached to NMSSI.

## Controlling Non-VTAM Terminals Through Customization Parameters

Terminals that are attached to the region through NMSSI can also be controlled through the ICS customization parameter group SSI. This parameter group provides a panel-driven interface for controlling non-VTAM terminal access (press F1 (Help) on the SSI Initialization Parameters panel for additional information). If NMSSI stops and restarts, you need to reapply the parameter group.

To avoid having to reapply the parameter group (that is, to ensure that the terminals always remain attached), you can update the NMSSI parameters by specifying a `CMD='ATTACH ...'` for each terminal. (See the section, *Commands*, on page D-10.)

## Remote Device System (RDS) Considerations

When using RDS to control allocations of devices to multiple systems, you need to modify the buffer size on the terminal device definition. The buffer size must be at least 65. For example, to attach device 04B to the NMSSI, specify the following:

```
DEFINE DEV ADD=04B, DEVTYPE=3779, BUF=65
```

## Using a Non-VTAM Terminal

A terminal that is connected through NMSSI displays the screen shown in Figure D-2 initially.

Figure D-2. Sample NMSSI Logo Screen

```
SOLV MVS1 Terminal: $LOCL4DF

      NN      NN MM      MM
      NNN     NN MMM     MMMM
      NNNN    NN MMMM   MM MM
      NN NN   NN MM MM MM MM
      NN NN   NN MM MMMM MM
      NN NN NN MM MM MM
      NN NNNN MM MM +-----+
      NN NNN MM MM / Version 5.0 /
      NN NN MM MM +-----+

      SSSSSS  SSSSSS  IIIIIIII
      SS      SS      II
      SS      SS      II
      SS      SS      II
      SSSSSS  SSSSSS  II
      SS      SS      II
      SS      SS      II
      SS      SS      II
      SSSSSS  SSSSSS  IIIIIIII
      Copyright (c) 1981,2001, Computer Associates International, Inc.

F1/13=Help F3/15=Detach F4/16=Detach Enter=Menu Other-PFK=Solve
```

This screen displays the NMSSI subsystem ID, the system name, the terminal name, and a list of available function keys. The function keys are described below.

### NMSSI Logo Screen Function Keys

#### **ENTER (Menu)**

Displays the NMSSI menu that lists the function keys for accessing the defined regions.

#### **F1 and F13 (Help)**

Displays online help for the non-VTAM terminal facility.

#### **F3, F4, F15, and F16 (Detach)**

Detaches the terminal from NMSSI. Use the ATTACH command to reconnect the terminal to NMSSI.

#### **F5 through F12 and F17 through F24 (Solve)**

Passes the terminal to the appropriate region for logon if a TERMACCESS parameter is defined for that function key. The terminal bypasses EASINET if it is active and displays the logon screen. At this point the terminal functions as if it were connected through VTAM.

## Accessing a Defined Region

You can access a defined region by pressing the appropriate function key from the NMSSI logo screen. You can also press ENTER to display the NMSSI menu that shows which function keys are defined and press the appropriate function key to access the region.

The NMSSI logo screen is displayed only when a terminal is first connected to NMSSI. Thereafter, the NMSSI menu is your NMSSI interface. Figure D-3 shows an example.

Figure D-3. NMSSI Menu

```
SOLV MVS1          *** NMSSI MENU ***          Terminal: $LOCL4DF
Pfk Nmid           Sysname  Status    Description
PF05 SOLVSNA1      MVS1     IN-SESSION SOLVE:OPS_SNA_1  *OUTPUT*
PF06 SOLVSNA2      MVS2     STARTING   SOLVE:OPS_SNA_2
PF07 -             -         -          -
PF08 -             -         -          -
PH09 -             -         -          -
PH10 -             -         -          -
PF11 SOLVMVS1      MVS1     IN-SESSION SOLVE:OPS_MVS_1  CURRENT
PF12 SOLVMVS2      MVS2     INACTIVE   SOLVE:OPS_MVS_2
PF17 -             -         -          -
PF18 -             -         -          -
PF19 -             -         -          -
PF20 -             -         -          -
PF21 -             -         -          -
PF22 -             -         -          -
PF23 -             -         -          -
PF24 -             -         -          -

F1/13=Help  F3/15=Logoff Current F4/16=Detach  Enter/Sysreq=Current
```

## Using the SYSREQ Key

Use the SYSREQ key to switch between a region and the NMSSI menu. By returning to the NMSSI menu, you can establish multiple sessions and access them any time by pressing the appropriate function key.

## Terminating a Session

Press F3 (Logoff Current) to terminate the session to the last accessed region.

To terminate another region, proceed as follows:

- Step 1. From the NMSSI menu, press the appropriate function key to access the region to make it current.
- Step 2. Press SYSREQ to return to the menu.
- Step 3. Press F3 to terminate the session.

## Session Status

Table D-1 describes the status of the sessions displayed on the NMSSI menu.

*Table D-1. Non-VTAM Terminal Sessions Status*

<b>Status</b>	<b>Description</b>
CONNECTED	The region is connected to an SSI region.
INACTIVE	The region is not connected to an SSI region.
IN-SESSION	The terminal has an active session with the region.
STARTING	The terminal is establishing a session with the region.

---

## Commands

Three new SSI commands are supplied for non-VTAM terminal support:

- ATTACH
- DETACH
- SHOW SSITERMS

These commands are issued by specifying the `CMD='command text'` parameter in the SSIPARMS member, or on the EXEC PARM statement. For information about these commands, see the *Command Reference*.

For information about other commands that are available for the SSI, see the chapter about the subsystem interface (SSI) in the *Management Services Administrator Guide*.

### Attaching Terminals—Examples

The following example attaches device 4DF:

```
ATTACH 4DF
```

The following example attaches device 4DF and assigns it the name TERMNAME. The accessed region is notified of the attachment, and the user's details are specified:

```
ATTACH 4DF NAME=TERMNAME AUTOLOG=YES DATA=USERID PASSWORD OPT
```

### Detaching Terminals Examples

The following example detaches the device 4DF:

```
DETACH 4DF
```

The following example detaches all devices:

```
DETACH ALL
```

---

## Implementing Non-VTAM Terminal Support by Using Telnet

Automation Services regions support a TCP/IP interface. You can customize that interface to support Telnet connections as follows:

- Step 1. Enter the **/ICS** shortcut at a **===>** prompt to list the region initialization parameter groups.
- Step 2. Enter **F TELNETSRVR** to find the TELNETSRVR parameter group.
- Step 3. Enter **U** beside the parameter group, and complete the fields as follows:

<b>Field</b>	<b>Value</b>
Allow TELNET Connections?	YES
Port 1	SHARED

Leave the other fields at their default values.

- Step 4. Press F6 (Action) to enable Telnet access.
- Step 5. Press F3 (File) to save the values so that Telnet access is enabled during region initialization.

After you have enabled Telnet access, users can use Telnet to access the region by using the port number specified in the Port No. for Inbound Connections field.

The parameter group allows you to specify up to five ports for Telnet access. For information about the parameters that you can use to customize Telnet access, see Help.

---

## Connecting to the Automation Services Region by Using Telnet

To access a region that supports Telnet connections, use the IP address and port number that have been set up.

To get the address and port number, enter **SHOW TCPIP** in the region to which you plan to connect.



---

## SMF Record Format

This chapter describes the format of System Management Facility (SMF) records that are written by Automation Services to store EventView, ResourceView and ServiceView statistics, and user-defined records.

**Note**

The following information also applies to SMS records on VOS3 systems.

---

## About Automation Services SMF Records

The following types of SMF records are written by Automation Services:

- EventView SMF record
- ResourceView and ServiceView SMF record
- User-defined record

The EventView SMF record has two subtypes; one is used to indicate the start of statistics collection, the second contains the actual data collected by EventView.

The ResourceView and ServiceView SMF record has only one subtype that contains data collected about resources or services. One record is written for each resource or service.

The user-defined record has a number of user-specified subtypes that contain data written by the SMFWRITE macro.

---

## SMF Header Format

All SMF records that are written by Automation Services have a header of the format shown in Table E-1.

*Table E-1. SMF Record Header Format*

<b>Offset</b>	<b>Position</b>		<b>Contains...</b>
<b>Dec.</b>	<b>Hex.</b>		
0	0	1-18	SMF record header
18	12	19-22	Subsystem identifier
22	16	23-24	The SMF record subtype
24	18	25-36	The region ID
36	24	37- <i>n</i>	The data to be written

---

## EventView SMF Record Format

The EventView SMF record that indicates the start of statistics collection has a subtype of 2000 (hexadecimal value). This subtype has no data associated with it.

The EventView SMF record that contains the EventView statistics data has a subtype of 2200 (hexadecimal value). All fields in the record are four bytes long.

Table E-2 lists the fields in this record.

*Table E-2. EventView Statistics SMF Record Format*

<b>Offset</b>	<b>Name</b>	<b>Length</b>	<b>Format</b>	<b>Description</b>	
<b>Dec.</b>	<b>Hex.</b>				
36	24	MSGCOUNT	4	BINARY	Number of messages processed
40	28	SUPPCOUNT	4	BINARY	Number of messages suppressed
44	2C	MODCOUNT	4	BINARY	Number of messages modified
48	30	REPCOUNT	4	BINARY	Number of replies issued to WTORs
52	34	SYSTEMCMD	4	BINARY	Number of system commands issued
56	38	SOLVECMD	4	BINARY	Number of MS commands issued
60	3C	SSOPRCSS	4	BINARY	Number of Automation Services processes issued
64	40	SSOCMD	4	BINARY	Number of Automation Services commands issued
68	44	VARSET	4	BINARY	Number of variables set
72	48	TMRPOP	4	BINARY	Number of timers that were triggered
76	4C	ISSUEMSG	4	BINARY	Number of messages issued

---

---

## ResourceView and ServiceView Record Format

The ResourceView and ServiceView SMF record has a subtype of 3000 (hexadecimal value).

Table E-3 shows the fields in this record.

*Table E-3. ResourceView and ServiceView SMF Record Format*

<b>Offset Dec.</b>	<b>Hex.</b>	<b>Name</b>	<b>Length</b>	<b>Format</b>	<b>Description</b>
36	24	SYSNAME	8	EBCDIC	System Image Name
44	2C	SYSVERSION	4	EBCDIC	System Image Version
48	30	RESCLASS	2	EBCDIC	Resource Class
50	32	NAME	18	EBCDIC	Resource or Service Name
68	44	AVIDATE	12	EBCDIC	Resource initialization date
80	50	AVITIME	12	EBCDIC	Resource initialization time
92	5C	AVDATE	4	BINARY	Date
96	60	AVTIME	12	EBCDIC	Time
108	6C	AVSTATUS	12	EBCDIC	Current status
120	78	AVATIME	4	BINARY	Total time available (in minutes)
124	7C	AVUTIME	4	BINARY	Total time unavailable (in minutes)
128	80	AVACOUNT	4	BINARY	Number of times the resource became available
132	84	AVUCOUNT	4	BINARY	Number of times the resource became unavailable
136	88	AVPMSGs	4	BINARY	Number of messages processed for the resource
140	8C	AVPCMDs	4	BINARY	Number of commands issued for the resource

---

---

## User-defined Record Format

A user-defined record SMF format has a subtype of 9xxx (hexadecimal value), where xxx is specified in the SMFWRITE macro.

The user defines the data fields in the record.

---

## Standardization of NCL Exits

The NCL exits provided by Automation Services are standardized to ensure that the following are constant:

- The storage of exit parameters in the knowledge base
- The substitution of variables in parameter strings for exits
- The passing of variables to exit code

This appendix describes:

- What an NCL exit is
- How to structure an NCL exit
- The various components of this structure

---

## What Is an NCL Exit Procedure?

An exit is a piece of user-written code that is executed at a predefined point. Numerous exit points are supplied by your product, to enable you to perform site-specific functions as required. For example, wherever you can specify a process in Automation Services, you can replace the process name with the name \$NCL and specify the name of an NCL exit to be called at that point. (\$NCL is documented in the *Automation Services Common User Guide*.)

---

## Standardized Structure

For ease of maintenance and debugging, an NCL exit should have the following structure (as discussed in this appendix):

- A commented introductory section
- A procedure that validates and interprets the parameters passed to the exit and converts them to variables
- The core processing code that performs the main function or functions of the exit
- A final piece of code that sets a return code and passes control back to the caller

A sample NCL exit procedure, \$RMPB06S, is supplied with Automation Services for your reference. This procedure creates a SOLVE:Problem record.

## Introducing the Exit

The commented introductory section should contain text that describes the following:

- The purpose of the exit
- The parameters passed to the exit
- The return codes set by the exit

## Parameter Processing

There are two types of parameters available to an NCL exit when it receives control (that is, starts executing):

- Parameters passed to the exit as NCL variables
- Parameters passed directly to the exit (that is, as a parameter string)

### Parameters Passed as NCL Variables

These parameters are created by the exit controller.

Different variables are passed to an exit, depending on the type of exit. For example, the variable `&ZRMDBNAME` (which contains a resource name) is passed to a state change exit. (Automation Services variables available to an NCL exit procedure are documented in the *Automation Services Common User Guide*.)

### Parameters Passed Directly to the Exit

These are in the form of a user-controlled parameter string that is stored, with the exit name, in the knowledge base. Parameters passed directly to the exit, such as the ones shown in the following example, are stored in the system variable `&ALLPARMS`:

```
CUSTACT=NOTIFY USER=FRED TEXT="Fred, the problem's fixed!"
```

Parameters passed directly to the exit can be in the keyword or the positional format (unless the exit is called by `$NCL`, in which case, positional parameters are not accepted), although the positional format is not recommended.

The following code fragment processes the parameters stored in the system variable `&ALLPARMS` and sets a return code. This code fragment can only process parameters that are in the *keyword* format:

```
&SETVARS PREFIX=PARM ERROR=CONTINUE DATA=&ALLPARMS
&IF &RETCODE NE 0 &THEN +
  &DO
    &SYSMSG = &STR &0-001 PARM ERROR MSG=&SYSMSG
    &RETURN &SYSMSG
  &DOEND
```

If the parameters are not in the keyword format, the system variable `&RETCODE` is set to eight, and an error message loaded into the system variable `&SYSMSG`. Using information from the `&SETVARS` statement, the `&DO` group of statements set an error message that:

- Identifies the exit procedure
- Supplies the identification number of the error within the exit
- Identifies the error condition

## Keyword Format

The standard keyword format is *KEYWORD=value*.

The keyword is a one to eight characters long, alphanumeric string. If the value contains blanks, then quotes are required, according to normal quoting rules. For example, a value of ACT IMM is specified as follows:

```
KEYWORD='ACT IMM'
```

A value of NOT ACT'D is specified in one of the following ways:

```
KEYWORD='NOT ACT ' 'D'  
KEYWORD="NOT ACT 'D"
```

If the initial &SETVARS statement is successful, keywords are converted into variables which have the format: *PARMKEYWORD*.

For example, imagine that the following parameters were passed:

```
CUSTACT=NOTIFY USER=FRED TEXT="Fred, the problem's fixed!"
```

Variables created from these parameters would be:

- PARMCUSTACT, containing the value: NOTIFY
- PARMUSER, containing the value: FRED
- PARMTEXT, containing the value: Fred, the problem's fixed!

## Positional Parameter Format

Positional parameters are named according to their position in a parameter string, such as &1, &2, and so on, and are separated by a space. In the given example, the variables created from the positional parameters would be:

- &1, containing the value: CUSTACT=NOTIFY
- &2, containing the value: USER=FRED
- &3, containing the value: TEXT="Fred,
- &4, containing the value: the
- &5, containing the value: problem's
- &6, containing the value: fixed!"

### Caution

If there are variables in a parameter string, and you are using the positional parameter format, this can produce unexpected results. For example, if the variable happens to have a null value, the remaining variables all shift across one argument. That is, if positional parameter &5 normally contains a variable value, then if that variable value is blank, &6 becomes positional parameter &5.

## **Main Processing Section**

This section of the exit contains the processing code that performs the main exit functions (that is, the functions for which reason the exit has been written.).

## **Exiting Back to the Caller**

After successful processing, the procedure sets the variable `&RETCODE` to zero, leaves `&SYSMSG` empty, and exits to the caller.

If processing is not successful, the variable `&RETCODE` must be set to eight, and a meaningful error message loaded into the variable `&SYSMSG`.



# G

---

## Using the Application Program Interface

This appendix describes the use and syntax of the following application programming interface (API) procedures:

\$RMCALL	\$RMDBAPI	\$RMEVENT	\$RMSTSET
\$RECALL	\$REDBAPI		

---

## Application Programming Interface Procedures

The application program interface enables sources external to products that use Automation Services to call Automation Services functions, and retrieve information about resources and services.

The following API procedures are supplied for this purpose:

- \$RMCALL—allows you to execute Automation Services commands, retrieve information about definitions in the knowledge base, retrieve resource and service status information, and delete extraneous link records.
- \$RMDBAPI—allows you to maintain the ResourceView definitions and, in a NetMaster for File Transfer region, file transfer rule sets and rules in the knowledge base.
- \$RMEVENT—allows you to send a message to a (typically user-defined) resource or service.
- \$RMSTSET—allows you to set the actual state of a defined resource or service.
- \$RECALL—allows you to process EventView variables from NCL procedures.
- \$REDBAPI—allows you to maintain the EventView definitions in the knowledge base.

The API can be called from:

- The message monitor or system console
- Batch programs
- NCL procedures
- State change exits

Each of the APIs and their calling conventions are described in the following sections.

**Note**

*Services* are considered to be a *specific type of resource*—that is, a resource with a class of SVC.

## About the \$RMCALL API

\$RMCALL is the main API procedure used to call Automation Services from external sources.

### \$RMCALL General Syntax

The general syntax for the \$RMCALL API procedure is shown below.

```
$RMCALL      OPT=SERVICE
              SERVICE=ACTION
              ACTION=[COMMAND | DBGET | PURGE | STGET]
              [ACBNAME=acb-name]
              [NAME=resource-name]
              [CLASS=cc]
              [SYSNAME=system-name]
              [VERSION=version]
              [COMMAND=command-name]
              [SYNC={YES | NO | NOTIFY}]
              [NCLID=ncl-id]
              [PARMS={'parm1=value1 parm2=value2 parm3=value3 ...'}]
```

You must specify the OPT=SERVICE and SERVICE=ACTION operands as shown. The ACTION operand is used to specify the function that the procedure performs. The remaining operands may be optional depending on the function being performed. Operands for each function are discussed in the section that describes the function.

If you code more than one value for an operand (for example, the PARMS operand) you must separate parameters with spaces and enclose them in quotation marks. Each option of the ACTION operand is described in the sections from page G-7 through page G-16.

## About the \$RMDBAPI API

\$RMDBAPI is the API procedure used to maintain ResourceView definitions from external sources. It does not support:

- Shared and sysplex system image definitions
- Sysplex class resource definitions
- NetMaster for SNA, NetMaster for TCP/IP, and NetSpy ResourceView definitions.

## \$RMDBAPI General Syntax

The general syntax for the \$RMDBAPI API procedure is shown below.

```
$RMDBAPI
  SERVICE={ACTIVATE|INACTIVATE|CREATE|DELETE|GET|LIST|SET}
  [TRUNCATE={YES|NO}]
  [{NAME=resource-name[MANNAME=manager-name]}
   {RSNAME=ft-ruleset-name[RMNAME=ft-rule-name]}]
  CLASS=cc
  [SYSNAME=system-name]
  [VERSION=version]
  [field-name-1=field value-1]
  [field-name-2=field value-2]
  .
  .
  .
  [field-name-n=field value-n]
```

The SERVICE operand is used to specify the function that the procedure performs. The remaining operands may be optional depending on the function being performed. Operands for each function are discussed in the section that describes the function.

Each option of the SERVICE operand is described in the sections from page G-18 through page G-41.

## About the \$RMEVENT API

\$RMEVENT is an API procedure used to send a message to a defined resource from external sources.

### \$RMEVENT General Syntax

The general syntax for the \$RMEVENT API procedure is shown below.

```
$RMEVENT  CLASS={ USRCLS | class-name }
           NAME=resource-name
           MSG='message-text'
```

Each option of the API is described in the section, *\$RMEVENT*, on page G-43.

## About the \$RMSTSET API

\$RMSTSET is an API procedure used to set the actual state of a defined resource from external sources.

### \$RMSTSET General Syntax

The general syntax for the \$RMSTSET API procedure is shown below.

```
$RMSTSET CLASS={USRCLS | class-name}  
          NAME=resource-name  
          STATUS=actual-state
```

Each option of the API is described in the section, *\$RMSTSET*, on page G-45.

## About the \$RECALL API

\$RECALL is an API procedure used to maintain EventView variables and control rule sets from external sources.

### \$RECALL General Syntax

The general syntax for the \$RECALL API procedure is shown below.

```
$RECALL SERVICE={ACTION|GET|SET}  
         [ACTION={ACT|INACT}]  
         [CLASS=VARIABLE]  
         NAME={'RULESET=rulesetname'|'VARNAME=variable-name'}  
         [PARMS='VALUE=new-value']  
         [DESC=value-description]
```

The SERVICE operand is used to specify the function that the procedure performs. The remaining operands may be optional depending on the function being performed. Operands for each function are discussed in the section that describes the function.

Each option of the SERVICE operand is described in the sections from page G-47 through page G-51.

## About the \$REDBAPI API

\$REDBAPI is the API procedure used to maintain EventView rule set and message rule definitions from external sources.

### \$REDBAPI General Syntax

The general syntax for the \$REDBAPI API procedure is shown below.

```
$REDBAPI SERVICE={CREATE|DELETE|GET|LIST}
          [TRUNCATE={YES|NO}]
          [NAME=rule-object-name]
          CLASS=cc
          [RULESET=ruleset-name]
          [RULEID=message-rule-id]
          [field-name-1=field value-1]
          [field-name-2=field value-2]
          .
          .
          .
          [field-name-n=field value-n]
```

The SERVICE operand is used to specify the function that the procedure performs. The remaining operands may be optional depending on the function being performed. Operands for each function are discussed in the section that describes the function.

Each option of the SERVICE operand is described in the sections from page G-53 through page G-65.

---

## \$RMCALL ACTION=COMMAND

### Syntax

```
$RMCALL      OPT=SERVICE
              SERVICE=ACTION
              ACTION=COMMAND
              [NAME=resource-name]
              [CLASS=cc]
              [SYSNAME=system-name]
              [VERSION=version]
              COMMAND=command-name
              SYNC={YES | NO | NOTIFY}
              [NCLID=ncl-id]
              [PARMS={'parm1=value1 parm2=value2 parm3=value3...'}]
```

### Use

Use this call to execute Automation Services commands.

### Operands

#### **OPT=SERVICE**

Indicates that an API service is to be performed.

#### **SERVICE=ACTION**

Indicates that an ACTION service is to be performed.

#### **ACTION=COMMAND**

Indicates that a command is to be processed.

#### **Note**

The following four operands (NAME, CLASS, SYSNAME, and VERSION) are included as operands in the API for the purpose of backward compatibility. Commands normally set these values, or prompt users for the values, depending on how the command is defined. If you do need to set these values (for example, you want to execute the command programmatically) set them through the PARMS operand. The values specified in the PARMS operand override the default values supplied by other operands.

#### **NAME=*resource-name***

The name of the resource to which the command applies.

#### **CLASS=*cc***

The two digit identifier of the class to which the resource belongs. (See Table 3-2 on page 3-36.)

**SYSNAME=*system-name***

The name of the system image where the command is executed.

**VERSION=*version***

The version of the system image where the command is executed.

**COMMAND=*command-name***

The name of the command to be executed.

**SYNC={YES|NO|NOTIFY}**

Specifies how the command is to be executed. The following values can be specified:

**YES**

The command will be executed synchronously: &RETCODE and &SYSMSG values are returned to the calling procedure.

**NO**

The command is executed asynchronously: results are not returned to the caller.

**NOTIFY**

The command is executed as for SYNC=NO but &SYSMSG values are returned to the caller's request queue in the form of \$\$\$MSG\$\$\$ *message-text*, where *message-text* is the contents of &SYSMSG.

**NCLID=*ncl-id***

When SYNC=NOTIFY is specified you need to specify an NCLID so that the value of &SYSMSG can be returned to the appropriate request queue.

**PARMS**={'*parm1=value1 parm2=value2 parm3=value3 ...*'}

Use this operand to specify parameters for the command. Delimit the parameters with spaces. If you specify more than one parameter you need to enclose the parameter list in quotation marks. The following parameters can be set when executing a command (see the note on page G-7):

**NAME**=*resource-name*

The name of the resource to which the command applies.

**CLASS**=*cc*

The two digit identifier of the class to which the resource belongs. (See Table 3-2 on page 3-36.)

**SYSNAME**=*system-name*

The name of the system image where the command is executed.

**VERSION**=*version*

The version of the system image where the command is executed.

For commands that require other parameters, see the section, *Supplied Commands That Require Parameters*, on page G-11.

## Returned Variable

**&SYSMSG**

Contains the message returned by \$RMCALL.

## Return Codes

The following return codes indicate the success or failure of command processing:

**&RETCODE**    **Meaning**

0	Processing was successful.
8	Processing failed.
16	Error occurred in call syntax.

## Examples

The following example shows how to execute the CHECKALL command on the local system:

```
&CALL      PROC=$RMCALL +
           PARS=( OPT=SERVICE , +
                 SERVICE=ACTION , +
                 ACTION=COMMAND , +
                 COMMAND=CHECKALL )
```

The following example assumes that you are using the SOLVE:Operations Automation product. It shows how to execute the ASA command against the printer (class 11) named RES001, located on version 0001 of the EASTPRD system:

```
&CALL      PROC=$RMCALL +
           PARS=( OPT=SERVICE , +
                 SERVICE=ACTION , +
                 ACTION=COMMAND , +
                 COMMAND=ASA , +
                 PARS=' SYSNAME=EASTPRD VERSION=0001 CLASS=11 NAME=RES001' )
```

## Supplied Commands That Require Parameters

Two commands, GLOBAL and LOAD, require parameters when they are executed programmatically. These parameters are described below.

### GLOBAL Command

**MODE={MANUAL|AUTOMATED}**

The global command allows you to specify whether the defined services and resources are to be controlled automatically or manually.

**Note**

If you specify AUTOMATED, resources and services that have MANUAL or IGNORED specified in their definitions are still controlled manually. If you specify MANUAL, then *all* elements are controlled manually.

### LOAD Command

**NEWSYS=*system-name***

Specifies the name of the new system image.

**NEWVERS=*version***

Specifies the version of the new system image.

**MODE={MANUAL|AUTOMATED}**

Sets the global mode, as documented in the previous section.

**WARM={YES|NO}**

Specifies whether a warm load or a cold load is to be performed.

For information about this feature, see the *Automation Services Common User Guide*.

---

## \$RMCALL ACTION=DBGET

### Syntax

```
$RMCALL    OPT=SERVICE
           SERVICE=ACTION
           ACTION=DBGET
           NAME=resource-name
           CLASS=cc
           SYSNAME=system-name
           VERSION=version
```

### Use

Use this call to retrieve information about a resource from the knowledge base.

If you make this call from an NCL procedure, ensure that you share the &ZRMDB-prefixed variables, for example, by adding the following statement before the call:

```
&CONTROL SHRVAR=( ZRMDB )
```

### Operands

#### **OPT=SERVICE**

Indicates that an API service is to be performed.

#### **SERVICE=ACTION**

Indicates that an ACTION service is to be performed.

#### **ACTION=DBGET**

Indicates that database information about a resource is to be retrieved.

#### **NAME=*resource-name***

The name of the resource for which information is to be retrieved.

#### **CLASS=*cc***

The two digit identifier of the class to which the resource belongs. (See Table 3-2 on page 3-36.)

#### **SYSNAME=*system-name***

The name of the system image where the resource is defined.

#### **VERSION=*version***

The version of the system image where the resource is defined.

## Returned Variables

### **&ZRMDB\***

Reference definition information about the specified resource is returned in &ZRMDB\* variables. For information about these variables, refer to the *Automation Services Common User Guide*.

### **&SYSMSG**

Contains the message returned by \$RMCALL.

## Return Codes

The following return codes indicate the success or failure of command processing:

<b>&amp;RETCODE</b>	<b>Meaning</b>
0	Processing was successful.
8	Processing failed.
16	Error occurred in call syntax.

## See Also

The \$RMDBAPI SERVICE=GET API on page G-35.

## Example

The following example retrieves information about the internal resource (class 21) named RES001, located on version 0001 of the EASTPRD1 system:

```
&CALL      PROC=$RMCALL +
           SHARE=( ZRMDB> ) +
           PARS=( OPT=SERVICE , +
                 SERVICE=ACTION , +
                 ACTION=DBGET , +
                 NAME=RES001 , +
                 CLASS=21 , +
                 SYSNAME=EASTPRD1 , +
                 VERSION=0001 )
```

---

## \$RMCALL ACTION=PURGE

### Syntax

\$RMCALL	OPT=SERVICE SERVICE=ACTION ACTION=PURGE ACBNAME= <i>acb-name</i>
----------	---

### Use

When a linked region is decommissioned without first unlinking it, extraneous link records are left behind in other regions that were connected to it. You can use this call to delete these extraneous link records.

### Operand

**ACBNAME=*acb-name***

The ACB name that identifies the link record.

### Returned Variable

**&SYSMMSG**

Contains the message returned by \$RMCALL.

### Return Codes

The following return codes indicate the success or failure of command processing:

**&RETCODE    Meaning**

0	Processing was successful.
8	Processing failed.
16	Error occurred in call syntax.

## Example

The following example deletes the SOLV1 link record:

```
&CALL      PROC=$RMCALL +  
           PARM=( OPT=SERVICE , +  
                 SERVICE=ACTION , +  
                 ACTION=PURGE , +  
                 ACBNAME=SOLV1 )
```

---

## \$RMCALL ACTION=STGET

### Syntax

```
$RMCALL    OPT=SERVICE
           SERVICE=ACTION
           ACTION=STGET
           NAME=resource-name
           CLASS=cc
           SYSNAME=system-name
           VERSION=version
```

### Use

Use this call to retrieve information about the current status of a resource.

If you make this call from an NCL procedure, ensure that you share the &ZRMST-prefixed variables, for example, by adding the following statement before the call:

```
&CONTROL SHRVAR=( ZRMST )
```

### Operands

#### **OPT=SERVICE**

Indicates that an API service is to be performed.

#### **SERVICE=ACTION**

Indicates that an ACTION service is to be performed.

#### **ACTION=STGET**

Indicates that status information about a resource is to be retrieved.

#### **NAME=*resource-name***

The name of the resource for which status information is to be retrieved.

#### **CLASS=*cc***

The two digit identifier of the class to which the resource belongs. (See Table 3-2 on page 3-36.)

#### **SYSNAME=*system-name***

The name of the system image where the resource is defined.

#### **VERSION=*version***

The version of the system image where the resource is defined.

## Returned Variables

### **&ZRMST\***

Status information about the specified resource is returned in &ZRMST\* variables. For information about these variables, refer to the *Automation Services Common User Guide*.

### **&SYSMSG**

Contains the message returned by \$RMCALL.

## Return Codes

The following return codes indicate the success or failure of resource status processing:

<b>&amp;RETCODE</b>	<b>Meaning</b>
0	Processing was successful.
8	Processing failed.
16	Error occurred in call syntax.

## Example

The following example retrieves status information about the internal resource (class 21) named RES001, located on version 0001 of the EASTPRD1 system:

```
&CALL PROC=$RMCALL +  
SHARE=( ZRMST> ) +  
PARMS=( OPT=SERVICE , +  
SERVICE=ACTION , +  
ACTION=STGET , +  
NAME=RES001 , +  
CLASS=21 , +  
SYSNAME=EASTPRD1 , +  
VERSION=0001 )
```

---

## \$RMDBAPI SERVICE=CREATE

### Syntax

```
$RMDBAPI SERVICE=CREATE
[TRUNCATE={YES|NO}]
[{NAME=resource-name[MANNAME=manager-name]}
 {RSNAME=ft-ruleset-name[RMNAME=ft-rule-name]]}
CLASS=cc
SYSNAME=system-name
VERSION=version
[field-name-1=field value-1]
[field-name-2=field value-2]
.
.
.
[field-name-n=field value-n]
```

### Use

Use this call to create a ResourceView definition in the knowledge base.

### Operands

#### Caution

Operand values must not contain the question mark (?) character.

#### Caution

If possible, do not use the semi-colon (;) in values. By default, a semi-colon is interpreted as a command separator if the API is executed (EXEC) or started (START).

If you must use the semi-colon and you are calling the API from an NCL procedure, use the &CALL verb.

#### **SERVICE=CREATE**

Indicates that a definition is to be created in the knowledge base.

#### **TRUNCATE={YES|NO}**

Specifies whether a field value should be truncated if it is longer than the field length.

- **YES** specifies that long values will be truncated and the definition will be created in the knowledge base.
- **NO** specifies that no truncation is allowed. If a long value is encountered, the definition will not be created in the knowledge base.

**NAME=*resource-name***

Specifies the name of the resource definition to be created (for example, the address of a DASD or the name of a started task).

This operand is not required if you are creating a system image, a file transfer ruleset, or a file transfer rule definition.

**MANNAME=*manager-name* (FT)**

Specifies the name of the manager that owns the resource, *resource-name* (for example, the definition of the file transfer manager that owns the monitor resource whose definition is to be maintained).

**RSNAME=*ft-ruleset-name* (FT)**

Specifies the name of the file transfer ruleset to which the specified rule, *ft-rule-name*, belongs.

**RMNAME=*ft-rule-name* (FT)**

Specifies the name of the file transfer rule definition to be created.

This operand is not required if you are creating a file transfer ruleset definition.

**CLASS=*cc***

Specifies the two digit identifier of the ResourceView class to which the definition belongs. (See Table 3-2 on page 3-36.)

**SYSNAME=*system-name***

Specifies the name of the system image to be created or the name of the system image where the specified resource is to be created.

**Note**

For file transfer rule sets and rules in a NetMaster for File Transfer region, the value of SYSNAME is FILTER.

**VERSION=*version***

Specifies the version of the system image.

**Note**

For file transfer rule sets and rules in a NetMaster for File Transfer region, the values of VERSION are 0006 and 0005 respectively.

***field-name-n=field-value-n***

Specifies the values of the fields in the definition.

For information about the field names, see the section, *ResourceView Definition Field Names*, on page G-21.

## Specifying Field Values When Calling \$RMDBAPI From an NCL Procedure

If you are making this call from an NCL procedure, consider the following:

- You can add the following statement before the call to share the field values:

```
&CONTROL SHRVAR=( ZRMDB )
```

Specify each value in `ZRMDBfield-name-n`. In this case, you do not need to specify the field name operands. You can, however, override the specified variable values by using the operands.

### Caution

If you share field values, the API ignores misspelled field names. If you misspell a name, the intended value will not be set.

- You can use the following statement to preserve case sensitive field values:

```
&CONTROL NOUCASE
```

- If you use the operands to pass the values, beware of the following:
  - The maximum length of an `&CALL` or `EXEC` statement is 2048 characters.
  - You *must not* enclose values in quotes. Use variables to assign strings.
  - If a value contains blanks, then for an `EXEC` statement, the procedure must use `&CONTROL NOVARSEG` and the value must be passed as a variable, for example:

```
&CONTROL NOVARSEG  
&DCMD=&STR D J,STC1  
EXEC $RMDBAPI SERVICE=CREATE ... DISPCMD=&DCMD ...
```

## Specifying Field Values When Submitting \$RMDBAPI as a Command

If a value contains blanks, enclose the value in quotes.

## Returned Variables

### &SYSMSG

Contains the message returned by \$RMDBAPI.

## Return Codes

The following return codes indicate the success or failure of the creation processing:

### **&RETCODE**   **Meaning**

0	Processing was successful.
4	Processing was successful, but truncation has occurred.
8	Processing failed.
16	Error occurred in call syntax.

## Example

The following example creates an EASTPRD1 version 2 system image definition (class 1), with truncation allowed:

```
&SDESC=&STR Eastern production system

&CALL   PROC=$RMDBAPI +
        PARS=( SERVICE=CREATE , +
              CLASS=01 , +
              SYSNAME=EASTPRD1 , +
              VERSION=0002 , +
              SDESC=&SDESC )
```

## ResourceView Definition Field Names

The following sections list the field names for ResourceView definitions. See also the product manuals for any product specific field names not included here.

The names are related to the corresponding field labels on the appropriate definition panels:

- Fields that are mandatory on a panel are mandatory in the API.
- Values that are valid in the panel fields are valid in the API.
- Fields that have default values inherit the values in the API.

System Image Fields

Table G-1 lists the system image field names that can be used in the \$RMDBAPI procedure.

*Table G-1. System Image Field Names Used in the \$RMDBAPI Procedure*

<b>Field Names</b>	<b>Field Label on Panel</b>
<b>System Image Definition</b>	
HOMESYS	Home System
SDESC	Short Description
LDESC1 to LDESC4	Long Description
RULEID	EventView Ruleset to Activate

## Resource Fields

Table G-2 lists the resource field names that can be used in the \$RMDBAPI procedure.

*Table G-2. Resource Field Names Used in the \$RMDBAPI Procedure*

<b>Field Names</b>	<b>Field Label on Panel</b>
<b>General Description</b>	
TYPE	Type
OWNRNME	CICS Region Name (classes 4 to 7)
OWNRCLS	CICS Region Class (classes 4 to 7)
REMONME	Remote Region Name (class 7)
REMOCLS	Class (class 7)
REMOSMF	SMFID (class 7)
CRFILT	Containment Filter
LUNAME	LU Name (classes 11 and 14)
VOLUME	Volume (class 16)
ACBNAME	ACB Name
MODE	Operation Mode
SDESC	Short Description
LDESC1 to LDESC4	Long Description
TEMPLAT	TemplateName
TMPLACT	Template action code (M, O, or R)
<b>Availability Map</b>	
SCHED	Map Name

*(Sheet 1 of 9)*

Table G-2. Resource Field Names Used in the \$RMDBAPI Procedure

Field Names	Field Label on Panel
<b>Group Filters (NMA)</b>	
RESNM1 to RESNM97	Resource Name
RESTP1 to RESTP97	Resource Type
FLTNM1 to FLTNM97	Filter Name
WGHT1 to WGHT97	Weight
WGHTT1 to WGHTT97	Weight Type
<b>State Thresholds (NMA)</b>	
UNKTHR	UNKNOWN
FAILTHR	FAILED
DEGDTHR	DEGRADED
STOPTHR	STOPPING
INACTHR	INACTIVE
STRTHR	STARTING
ACTTHR	ACTIVE
NOKTHR	Not OK

(Sheet 2 of 9)

Table G-2. Resource Field Names Used in the \$RMDBAPI Procedure

Field Names	Field Label on Panel
<b>Activation Details</b>	
INITCMD	System Command
INIMSGT	Expected Activation Completion Message
INx	See the panels, <i>Define Extended Filter Definitions</i> , on page G-30 to, <i>Define Event Documentation</i> , on page G-31
INIPNAM	ProcessName
INIPRC1 to INIPRC2	Optional Parameters
INITIME	Timeout After
INITMST	On Timeout Assume Status of
<b>Restart Control Parameters</b>	
RETRYLM	Retry Attempt Limit
RETRYTM	Retry Time Limit
RETRYCM	System Command
RETRYPR	ProcessName
RETRYP1 to RETRYP2	Optional Parameters

(Sheet 3 of 9)

Table G-2. Resource Field Names Used in the \$RMDBAPI Procedure

Field Names	Field Label on Panel
<b>Inactivation Details</b>	
TRMTCMD	System Command
TRMMSGT	Expected Inactivation Completion Message
TRx	See the panels, <i>Define Extended Filter Definitions</i> , on page G-30 to, <i>Define Event Documentation</i> , on page G-31
TRMPNAM	ProcessName
TRMPRC1 to TRMPRC2	Optional Parameters
TRMTIME	Timeout After
TRMTMST	On Timeout Assume Status of
TRMFRET	Try Force Inactivation
<b>Force Inactivation Details</b>	
FTRMCMD	System Command
FTRMSGT	Expected Force Inactivation Completion Message
FTx	See the panels, <i>Define Extended Filter Definitions</i> , on page G-30 to, <i>Define Event Documentation</i> , on page G-31
FTRPNAM	ProcessName
FTRMPR1 to FTRMPR2	Optional Parameters
FTRTIME	Timeout After
FTRTMST	On Timeout Assume Status of
<b>Display and Heartbeat Details</b>	
DISPCMD	System Command
HBEATIN	Heartbeat Interval

(Sheet 4 of 9)

Table G-2. Resource Field Names Used in the \$RMDBAPI Procedure

Field Names	Field Label on Panel
<b>Display and Heartbeat Details</b> (continued)	
ACTMSG	Message Text (ACTIVE status)
DAx	For ACTIVE status (see the panels, <i>Define Extended Filter Definitions</i> , on page G-30 to, <i>Define Event Documentation</i> , on page G-31)
INACMSG	Message Text (INACTIVE status)
DIx	For INACTIVE status (see the panels, <i>Define Extended Filter Definitions</i> , on page G-30 to, <i>Define Event Documentation</i> , on page G-31)
STRMSG	Message Text (STARTING status)
DSx	For STARTING status (see the panels, <i>Define Extended Filter Definitions</i> , on page G-30 to, <i>Define Event Documentation</i> , on page G-31)
STOPMSG	Message Text (STOPPING status)
DPx	For STOPPING status (see the panels, <i>Define Extended Filter Definitions</i> , on page G-30 to, <i>Define Event Documentation</i> , on page G-31)
DEGMSG	Message Text (DEGRADED status)
DDx	For DEGRADED status (see the panels, <i>Define Extended Filter Definitions</i> , on page G-30 to, <i>Define Event Documentation</i> , on page G-31)
FAILMSG	Message Text (FAILED status)
DFx	For FAILED status (see the panels, <i>Define Extended Filter Definitions</i> , on page G-30 to, <i>Define Event Documentation</i> , on page G-31)
UNKMSG	Message Text (UNKNOWN status)
DUx	For UNKNOWN status (see the panels, <i>Define Extended Filter Definitions</i> , on page G-30 to, <i>Define Event Documentation</i> , on page G-31)

(Sheet 5 of 9)

Table G-2. Resource Field Names Used in the \$RMDBAPI Procedure

Field Names	Field Label on Panel
<b>Display and Heartbeat Details</b> <i>(continued)</i>	
DISPNAM	ProcessName
DISPRC1	Optional Parameters
<b>Status Monitor Message Details</b>	
MONMT01 to MONMT97	Messages
M01x to M97x	See the panels, <i>Define Extended Filter Definitions</i> , on page G-30 to, <i>Define Event Documentation</i> , on page G-31
MONPR01 to MONPR97	Pty
MONST01 to MONST97	Status
<b>State Change Exits</b>	
SBAPROC	Process (before activation)
SBAPRM1 and SBAPRM2	Parameters (before activation)
STYPE1 to STYPE12	State Type
SFROM1 to SFROM12	Change From
STO1 to STO12	Change To
SPROC1 to SPROC12	Process
SPRM11 to SPRM112	Parameters (first line)
SPRM21 to SPRM212	Parameters (second line)

(Sheet 6 of 9)

Table G-2. Resource Field Names Used in the \$RMDBAPI Procedure

Field Names	Field Label on Panel
<b>Automation Log Details</b>	
LOGTSIZ	Log Table Size
LOGAUTO	Log to Automation Log
LOGMVSC	Log to Console
LOGOCS	Log to OCS Window
LOGSYS	Log All System Msgs
LOGAUDT	Log Internal Audit Trail
<b>Owner Details</b>	
OWNNME1 and OWNNME2	Name
OWNUID1 and OWNUID2	Userid
OWNGRP1 and OWNGRP2	Group
OWNPHB1 and OWNPHB2	Phone Business Hours
OWNPHA1 and OWNPHA2	Phone After Hours
OWNPGR1 and OWNPGR2	Pager Number
<b>Extended Function Exit</b>	
FUNCNAM	Function Name
PARAM1 to PARAM12	Parameters

(Sheet 7 of 9)

Table G-2. Resource Field Names Used in the \$RMDBAPI Procedure

Field Names	Field Label on Panel
<b>Define Extended Filter Definitions</b>	
xWILD	Wildcard Character
xSP1 to xSP5	Strt Pos
xWN1 to xWN5	Word Num
xOP1 to xOP5	Opr
xSC1 to xSC5	Scan Text
xEXP1 to xEXP5 (combined)	<i>strt-pos,word-num,opr,scan-text</i>
xRID	Expression
<b>Define Event Related Actions</b>	
xMOD	Mode
xICMD	System Command
xRPLY	Reply (if WTOR)
xGALT	Generate Rsc Event
xLTXT	Log Message
xXPNM	ProcessName
xXPR1 and xXPR2	Optional Parameters

(Sheet 8 of 9)

Table G-2. Resource Field Names Used in the \$RMDBAPI Procedure

Field Names	Field Label on Panel
<b>Define Event Exits</b>	
xSTXP	Process (State Change)
xSTX1 and xSTX2	Parameters (State Change)
xPRXP	Process (Problem)
xPRX1 and xPRX2	Parameters (Problem)
xGEXP	Process (General)
xGEX1 and xGEX2	Parameters (General)
<b>Define Extended Display Attribute</b>	
xEVDS	Extended Display (EXTDISP)
xINTN	Intensity
xCOLR	Color
xHLIT	Highlight
xICON	Use on Graphic Monitor?
xSEV	Severity
xFKWD	Keyword Value
xKWD1 to xKWD6	Var
xVAL1 to xVAL6	Value
<b>Define Event Documentation</b>	
xTMSG	Target Message
xNT1 to xNT12	Notes
<i>(Sheet 9 of 9)</i>	

---

## \$RMDBAPI SERVICE=DELETE

### Syntax

```
$RMDBAPI SERVICE=DELETE
          [{NAME=resource-name[MANNAME=manager-name]}]
          {RSNAME=ft-ruleset-name[RMNAME=ft-rule-name]}]
          CLASS=cc
          SYSNAME=system-name
          VERSION=version
```

### Use

Use this call to delete a ResourceView definition from the knowledge base.

### Operands

#### **SERVICE=DELETE**

Indicates that a definition is to be deleted from the knowledge base.

#### **NAME=*resource-name***

Specifies the name of the resource definition to be deleted.

You cannot delete a resource that owns dependent resources (for example, a CICS started task that owns CICS resources).

This operand is not required if you are deleting a system image, a file transfer ruleset, or a file transfer rule.

#### **MANNAME=*manager-name* (FT)**

Specifies the name of the manager that owns the resource, *resource-name* (for example, the definition of the file transfer manager that owns the monitor resource whose definition is to be deleted).

#### **RSNAME=*ft-ruleset-name* (FT)**

Specifies the name of the file transfer ruleset to be deleted or the name of the file transfer ruleset from which the specified rule, *ft-rule-name*, is to be deleted.

You cannot delete an active ruleset.

#### **Caution**

When you delete a file transfer ruleset, you also delete the rules it owns.

**RMNAME=*ft-rule-name*** (FT)

Specifies the name of the file transfer rule definition to be deleted.

This operand is not required if you are deleting a file transfer ruleset.

**CLASS=*cc***

Specifies the two digit identifier of the ResourceView class to which the definition belongs. (See Table 3-2 on page 3-36.)

**SYSNAME=*system-name***

Specifies the name of the system image to be deleted or the name of the system image from which the specified resource is to be deleted.

**Caution**

When you delete a system image, you also delete the resources it owns.

**Note**

For file transfer rule sets and rules in a NetMaster for File Transfer region, the value of SYSNAME is FILTER.

**VERSION=*version***

Specifies the version of the system image.

**Note**

For file transfer rule sets and rules in a NetMaster for File Transfer region, the values of VERSION are 0006 and 0005 respectively.

**Returned Variables**

**&SYMSG**

Contains the message returned by \$RMDBAPI.

**Return Codes**

The following return codes indicate the success or failure of the deletion processing:

<b>&amp;RETCODE</b>	<b>Meaning</b>
0	Processing was successful.
8	Processing failed.
16	Error occurred in call syntax.

## Example

The following example deletes the RES001 printer definition (class 11) from the EASTPRD1 version 1 system image:

```
&CALL      PROC=$RMDBAPI  +
           PARS=( SERVICE=DELETE , +
                 NAME=RES001 , +
                 CLASS=11 , +
                 SYSNAME=EASTPRD1 , +
                 VERSION=0001 )
```

---

## \$RMDBAPI SERVICE=GET

### Syntax

```
$RMDBAPI    SERVICE=GET
             [{NAME=resource-name[MANNAME=manager-name]}|
              {RSNAME=ft-ruleset-name[RMNAME=ft-rule-name]}]
             CLASS=cc
             SYSNAME=system-name
             VERSION=version
```

### Use

Use this call to retrieve information about a ResourceView definition in the knowledge base.

If you make this call from an NCL procedure, ensure that you share the &ZRMDB-prefixed variables, for example, by adding the following statement before the call:

```
&CONTROL SHRVAR=( ZRMDB )
```

### Operands

#### **SERVICE=GET**

Indicates that information about a definition is to be retrieved from the knowledge base.

#### **NAME=*resource-name***

Specifies the name of the resource definition for which information is to be retrieved.

This operand is not required if you are retrieving information about a system image definition, a file transfer ruleset, or a file transfer rule.

#### **MANNAME=*manager-name* (FT)**

Specifies the name of the manager that owns the resource, *resource-name* (for example, the definition of the file transfer manager that owns the monitor resource for which information is to be retrieved).

**RSNAME=*ft-ruleset-name*** (FT)

Specifies the name of the file transfer ruleset to which the specified rule, *ft-rule-name*, belongs.

**RMNAME=*ft-rule-name*** (FT)

Specifies the name of the file transfer rule definition for which information is to be retrieved.

This operand is not required if you are retrieving information about a file transfer ruleset definition.

**CLASS=*cc***

Specifies the two digit identifier of the ResourceView class to which the definition belongs. (See Table 3-2 on page 3-36.)

**SYSNAME=*system-name***

Specifies the name of the system image for which information is to be retrieved or the name of the system image that owns the resource for which information is to be retrieved.

**Note**

For file transfer rule sets and rules in a NetMaster for File Transfer region, the value of SYSNAME is FILTER.

**VERSION=*version***

Specifies the version of the system image.

**Note**

For file transfer rule sets and rules in a NetMaster for File Transfer region, the values of VERSION are 0006 and 0005 respectively.

## Returned Variables

**&ZRMDB*field-name***

Knowledge base information about the specified resource is returned in &ZRMDB*field-name* variables. For information about the field names, see the section, *ResourceView Definition Field Names*, on page G-21.

**&SYSMSG**

Contains the message returned by \$RMDBAPI.

## Return Codes

The following return codes indicate the success or failure of the retrieval processing:

<b>&amp;RETCODE</b>	<b>Meaning</b>
0	Processing was successful.
8	Processing failed.
16	Error occurred in call syntax.

## Example

The following example retrieves information about the RES001 internal resource (class 21), located on the EASTPRD1 version 1 system image:

```
&CALL      PROC=$RMDBAPI +  
           SHARE=( ZRMDB> ) +  
           PARS=( SERVICE=GET , +  
                 NAME=RES001 , +  
                 CLASS=21 , +  
                 SYSNAME=EASTPRD1 , +  
                 VERSION=0001 )
```

---

## \$RMDBAPI SERVICE=LIST

### Syntax

```
$RMDBAPI SERVICE=LIST  
          [RSNAME=ft-ruleset-name]  
          CLASS=cc  
          [SYSNAME=system-name]  
          [VERSION=version]
```

### Use

Use this call to list ResourceView definitions in the knowledge base.

If you make this call from an NCL procedure, ensure that you share the &ZRMLST-prefixed variables, for example, by adding the following statement before the call:

```
&CONTROL SHRVAR=( ZRMLST )
```

## Operands

### **SERVICE=LIST**

Indicates that selected definitions in the knowledge base are to be listed in `&ZRMLSTnnnn` variables.

### **RSNAME=*ft-ruleset-name* (FT)**

Specifies the name of the file transfer ruleset that owns the rules to be listed.

If you want to list the rulesets, do not specify this operand.

### **CLASS=*cc***

Specifies the two digit identifier of the ResourceView class to which the definitions belong. (See Table 3-2 on page 3-36.)

### **SYSNAME=*system-name***

Specifies the name of the system image that owns the resource definitions to be listed.

#### **Note**

For file transfer rule sets and rules in a NetMaster for File Transfer region, the value of SYSNAME is FILTER.

### **VERSION=*version***

Specifies the version of the system image.

#### **Note**

For file transfer rule sets and rules in a NetMaster for File Transfer region, the values of VERSION are 0006 and 0005 respectively.

## Returned Variables

### **&ZRMLSTnnnn**

Each knowledge base definition entry is returned in an `&ZRMLSTnnnn` variable.

### **&SYSMSG**

Contains the message returned by \$RMDBAPI.

## Return Codes

The following return codes indicate the success or failure of the list processing:

<b>&amp;RETCODE</b>	<b>Meaning</b>
0	Processing was successful.
8	Processing failed.
16	Error occurred in call syntax.

## Example

The following example retrieves the list of all versions of the EASTPRD1 system image:

```
&CALL PROC=$RMDBAPI +  
SHARE=(ZRMLST>) +  
PARMS=(SERVICE=LIST,+  
CLASS=01,+  
SYSNAME=EASTPRD1)
```

---

## \$RMDBAPI SERVICE=SET

### Syntax

```
$RMDBAPI    SERVICE=SET
             [TRUNCATE={YES|NO}]
             CLASS=01
             SYSNAME=system-name
             VERSION=version
             [field-name-1=field value-1]
             [field-name-2=field value-2]
             .
             .
             .
             [field-name-n=field value-n]
```

### Use

Use this call to change field values in a system image definition. The SET function is not available to other definitions in the knowledge base.

### Operands

#### **SERVICE=SET**

Indicates that changes are to be made in the specified system image definition.

#### **TRUNCATE={YES|NO}**

Specifies whether a field value should be truncated if it is longer than the field length.

- **YES** specifies that long values will be truncated and the definition will be changed.
- **NO** specifies that no truncation is allowed. If a long value is encountered, the definition will not be changed.

#### **CLASS=01**

Indicates that a system image definition is to be changed.

#### **SYSNAME=*system-name***

Specifies the name of the system image definition to be changed.

#### **VERSION=*version***

Specifies the version of the system image definition.

***field-name-n=field-value-n***

Specifies the values of the fields to be changed in the definition.

You cannot delete a field value.

For information about the field names, see the section, *ResourceView Definition Field Names*, on page G-21. For things to consider when you specify field values, see the sections, *Specifying Field Values When Calling \$RMDBAPI From an NCL Procedure*, on page G-20 and, *Specifying Field Values When Submitting \$RMDBAPI as a Command*, on page G-20.

## Returned Variables

**&SYMSMSG**

Contains the message returned by \$RMDBAPI.

## Return Codes

The following return codes indicate the success or failure of the change processing:

**&RETCODE    Meaning**

0	Processing was successful.
4	Processing was successful, but truncation has occurred.
8	Processing failed.
16	Error occurred in call syntax.

## Example

The following example changes the value of the fourth line of the Long Description field in the EASTPRD1 version 2 system image definition (class 1), with truncation allowed:

```
&LDESC4=&STR (Upgrade in progress)
```

```
&CALL    PROC=$RMDBAPI +  
          PARMS=( SERVICE=SET , +  
                  CLASS=01 , +  
                  SYSNAME=EASTPRD1 , +  
                  VERSION=0002 , +  
                  LDESC4=&LDESC4 )
```

---

## \$RMEVENT

### Syntax

```
$RMEVENT CLASS={ USRCLS | class-name }  
          NAME=resource-name  
          MSG='message-text'
```

### Use

Use the \$RMEVENT API procedure to send a message to a defined resource. The resource definition must interpret the message and control the actual state of the resource. If necessary, relevant actions are invoked as part of this process. This procedure is typically used to control user-defined resources.

### Operands

**CLASS**={ USRCLS | *class-name* }

The class of the resource that is the subject of the message. The default, USRCLS, specifies a user-defined resource class.

**NAME**=*resource-name*

The name of the resource that is the subject of the message.

**MSG**='message-text'

The text of the message that is sent to the resource. The message text must be enclosed in quotation marks.

### Returned Variable

**&SYSMSG**

Contains the message returned by \$RMEVENT.

### Return Codes

The following return codes indicate the success or failure of processing:

**&RETCODE**    **Meaning**

0	Processing was successful.
8	Processing failed.
16	Error occurred in call syntax.

## Example

The following example sends a user-defined message (that is, a message defined in the `USRCLS` class) to the resource named `MYRES01`:

```
$RMEVENT NAME=MYRES01 MSG='RECOVER'
```

---

## \$RMSTSET

### Syntax

```
$RMSTSET CLASS={USRCLS | class-name}  
          NAME=resource-name  
          STATUS=actual-state
```

### Use

Use this API to set the actual state of a defined resource.

### Operands

**CLASS={ USRCLS | *class-name* }**

The class of the resource that is the subject of the status change. USRCLS specifies the user-defined resource class.

**NAME=*resource-name***

The resource that is the subject of the status change.

**STATUS=*actual-state***

The actual state to which the resource is to be set.

### Returned Variable

**&SYMSG**

Contains the message returned by \$RMSTSET.

### Return Codes

The following return codes indicate the success or failure of command processing:

**&RETCODE    Meaning**

0	Processing was successful.
8	Processing failed.
16	Error occurred in call syntax.

## Example

The following example sets the actual state of MYRES01 to FAILED:

```
$RMSTSET NAME=MYRES01 STATUS=FAILED
```

---

## \$RECALL SERVICE=SET

### Syntax

\$RECALL	SERVICE=SET CLASS=VARIABLE NAME='VARNAME= <i>variable-name</i> ' PARMS='VALUE= <i>new-value</i> ' [DESC= <i>value-description</i> ]
----------	---

### Use

Use this API to set the value of an EventView variable.

### Operands

#### **SERVICE=SET**

A value is to be set.

#### **CLASS=VARIABLE**

The value to be set is an EventView variable.

#### **NAME='VARNAME=*variable-name*'**

Identifies the EventView variable. *variable-name* can be up to eight characters long.

#### **PARMS='VALUE=*new-value*'**

The new value to be set for the specified variable.

#### **DESC=*value-description***

Optional entry to describe the variable value.

### Returned Variable

#### **&SYMSG**

Contains the message returned by \$RECALL.

## Return Codes

The following return codes indicate the success or failure of command processing:

<b>&amp;RETCODE</b>	<b>Meaning</b>
0	Processing was successful.
8	Processing failed.
16	Error occurred in call syntax.

## Example

The following example sets the value of TEST:

```
&VALUE = &ZQUOTE Test's value at &TIME
&DESC = &ZQUOTE Why Test was changed
&PARMS = &ZQUOTE VALUE=&VALUE DESC=&DESC
&CALL PROC=$RECALL +
    PARS=( SERVICE=SET, +
           CLASS=VARIABLE, +
           NAME= 'VARNAME=TEST' , +
           PARS=&PARMS )
```

---

## \$RECALL SERVICE=GET

### Syntax

\$RECALL	SERVICE=GET CLASS=VARIABLE NAME='VARNAME= <i>variable-name</i> '
----------	--

### Use

Use this API to retrieve the value of an EventView variable.

If you make this call from an NCL procedure, ensure that you share the &\$REVAR-prefixed variables, for example, by adding the following statement before the call:

```
&CONTROL SHRVAR=( $REVAR )
```

### Operands

#### **SERVICE=GET**

A value is to be retrieved.

#### **CLASS=VARIABLE**

The value to get is the value of an EventView variable.

#### **NAME='VARNAME=*variable-name*'**

Identifies the EventView variable.

## Returned Variables

### **&\$REVARNAME**

Contains the name of the variable.

### **&\$REVARVALUE**

Contains the value of the variable.

### **&\$REVARDESC**

Contains a description of the value of the variable.

### **&\$REVARSTATS**

Contains the date, time, and the user ID of the user who last updated the value.

### **&SYMSG**

Contains the message returned by \$RECALL.

## Return Codes

The following return codes indicate the success or failure of command processing:

<b>&amp;RETCODE</b>	<b>Meaning</b>
0	Processing was successful.
8	Processing failed.
16	Error occurred in call syntax.

## Example

The following example retrieves the value of TEST:

```
&CALL PROC=$RECALL SHARE=( $RE> ) +  
      PARMS=( SERVICE=GET , +  
              CLASS=VARIABLE , +  
              NAME=' VARNAME=TEST ' )  
&WRITE ZREVTEMP=&$REVARVALUE
```

---

## \$RECALL SERVICE=ACTION

### Syntax

```
$RECALL    SERVICE=ACTION
           ACTION={ACT | INACT}
           NAME='RULESET=rulesetname'
```

### Use

Use this API to activate or inactive a ruleset.

### Operands

#### **SERVICE=ACTION**

An action is to be performed on a ruleset.

#### **ACTION={ ACT | INACT }**

The ruleset is to be activated or inactivated.

#### **NAME='RULESET=*rulesetname*'**

Names the ruleset to be activated or inactivated.

### Returned Variables

#### **&SYSMSG**

Contains the message returned by \$RECALL.

### Return Codes

The following return codes indicate the success or failure of command processing:

#### **&RETCODE    Meaning**

0	Processing was successful.
8	Processing failed.
16	Error occurred in call syntax.

## Examples

The following example activates ruleset SET01:

```
$RECALL SERVICE=ACTION ACTION=ACT NAME='RULESET=SET01'
```

The following example inactivates ruleset SET01:

```
$RECALL SERVICE=ACTION ACTION=INACT NAME='RULESET=SET01'
```

---

## \$REDBAPI SERVICE=CREATE

### Syntax

```
$REDBAPI    SERVICE=CREATE
            [TRUNCATE={YES|NO}]
            CLASS=cc
            RULESET=ruleset-name
            [field-name-1=field value-1]
            [field-name-2=field value-2]
            .
            .
            .
            [field-name-n=field value-n]
```

### Use

Use this call to create an EventView rule set or message rule definition in the knowledge base.

### Operands

#### Caution

If possible, do not use the semi-colon (;) in values. By default, a semi-colon is interpreted as a command separator if the API is executed (EXEC) or started (START).

If you must use the semi-colon and you are calling the API from an NCL procedure, use the &CALL verb.

#### **SERVICE=CREATE**

Indicates that a definition is to be created in the knowledge base.

#### **TRUNCATE={YES|NO}**

Specifies whether a field value should be truncated if it is longer than the field length.

- **YES** specifies that long values will be truncated and the definition will be created in the knowledge base.
- **NO** specifies that no truncation is allowed. If a long value is encountered, the definition will not be created in the knowledge base.

#### **CLASS=*cc***

Specifies the two digit identifier of the EventView class to which the definition belongs as follows: 93 for rulesets and 94 for message rules.

**RULESET=*ruleset-name***

Specifies the name of the ruleset to be created or the name of the ruleset where a message rule is to be created.

***field-name-n=field-value-n***

Specifies the values of the fields in the definition.

A value must not contain the question mark (?) character.

For information about the field names, see the section, *EventView Definition Field Names*, on page G-55.

## Specifying Field Values When Calling \$REDBAPI From an NCL Procedure

If you are making this call from an NCL procedure, consider the following:

- You can add the following statement before the call to share the field values:

```
&CONTROL SHRVAR=( ZRE )
```

Specify each value in ZRE*field-name-n*. In this case, you do not need to specify the field name operands. You can, however, override the specified variable values by using the operands.

**Caution**

If you share field values, the API ignores misspelled field names. If you misspell a name, the intended value will not be set.

- You can use the following statement to preserve case sensitive field values:

```
&CONTROL NOUCASE
```

- If you use the operands to pass the values, beware of the following:

- The maximum length of an &CALL or EXEC statement is 2048 characters.
- You *must not* enclose values in quotes. Use variables to assign strings.
- If a value contains blanks, then for an EXEC statement, the procedure must use &CONTROL NOVARSEG and the value must be passed as a variable, for example:

```
&CONTROL NOVARSEG  
&DESC=&STR SUPPRESSION RULES  
EXEC $REDBAPI SERVICE=CREATE ... SDESC=&DESC ...
```

## Specifying Field Values When Submitting \$REDBAPI as a Command

If a value contains blanks, enclose the value in quotes.

## Returned Variables

### **&ZRENAME**

Contains the name of a created message rule.

### **&SYSMMSG**

Contains the message returned by \$REDBAPI.

## Return Codes

The following return codes indicate the success or failure of the creation processing:

### **&RETCODE**    **Meaning**

0	Processing was successful.
4	Processing was successful, but truncation has occurred.
8	Processing failed.
16	Error occurred in call syntax.

## Example

The following example creates the SUPP ruleset definition (class 93), with truncation allowed:

```
&SDESC=&STR Suppression rules

&CALL    PROC=$REDBAPI +
          PARS=( SERVICE=CREATE , +
                 CLASS=93 , +
                 RULESET=SUPP , +
                 SDESC=&SDESC )
```

## EventView Definition Field Names

The following sections list the field names for EventView rulesets and message rules.

The names are related to the corresponding field labels on the appropriate definition panels:

- Fields that are mandatory on a panel are mandatory in the API
- Values that are valid in the panel fields are valid in the API.
- Fields that have default values inherit the values in the API.

## Ruleset Fields

Table G-3 lists the rule set field names that can be used in the \$REDBAPI procedure.

*Table G-3. EventView Ruleset Field Names Used in the \$REDBAPI Procedure*

<b>Field Names</b>	<b>Field Label on Panel</b>
<b>Ruleset Description</b>	
RULSTAT	Ruleset Status
SDESC	Short Description
RSDELIV	Default Message Delivery
RSMOD	Perform Message Modification?
RSACT	Perform Action?
RSLOG	Log Ruleset Activity?
RSSTAT	Collect Statistics?
RSLEARN	Learn New Messages?
<b>Ruleset Comments</b>	
COMMENT1 to COMMENT12	Comment Text

## Message Rule Fields

Table G-4 lists the message rule field names that can be used in the \$REDBAPI procedure.

*Table G-4. EventView Message Rule Field Names Used in the \$REDBAPI Procedure*

<b>Field Names</b>	<b>Field Label on Panel</b>
<b>Message Filter</b>	
RULSTAT	Rule Status
SDESC	Short Description
TSTTXT	Message Text
JOBNAME	Job Name
EJOBTYPE	Job Type
RULEPRI	Rule Priority
BESTFIT	Execute if not Best Fit?
DAYMAP (xxxxxxx) or DAY1 to DAY7	On Days
TSTART1 to TSTART2	Rangen Start
TEND1 to TEND2	Rangen End

*(Sheet 1 of 4)*

Table G-4. EventView Message Rule Field Names Used in the \$REDBAPI Procedure

Field Names	Field Label on Panel
<b>Message Filter</b> (continued)	
<b>Extended Message Filter</b>	
EWILDC	Wildcard Character
EDESCCD	Descriptor Code
EROUTCD	Route Code
EMSGID	Message ID
MVSSYS	System Name
ESTPOS1 to ESTPOS5	Strt Pos
EWORD1 to EWORD5	Word Num
EOPER1 to EOPER5	Opr
ETXT1 to ETXT5	Scan Text
EEXPR	Expression
<b>Set Test Variables</b>	
TSTVAR1 to TSTVAR6 ( <i>name=value</i> ) or TVAR1 and TVALUE1 to TVAR6 and TVALUE6	Name and Value
<b>Message Delivery</b>	
MDELIV	Deliver
THRSMAX	Maximum Number
THRSINT	Time Interval
THRSACT	Do Action
THRSCOR	Correlation Key

(Sheet 2 of 4)

Table G-4. *EventView Message Rule Field Names Used in the \$REDBAPI Procedure*

<b>Field Names</b>	<b>Field Label on Panel</b>
<b>Message Modification</b>	
MSGTXT	Replacement Text
MDESCCD	Set Descriptor Code
MROUTCD	Set Route Code
COLOR	Color
HLITE	Highlight
INTENS	Intensity
MON	Monitor?
ALARM	Alarm?
NRD	NRD?
MSGCODE	Message Code
<b>Message Actions</b>	
REPLTXT	Reply Text
SYSCMD	System Command
SOLVCMD	MS Command
SSOPROC (combined) or SSOPNAME and SSOPPRM	Process and Parameters
SSOCMD (combined) or SSOCNAME and SSOCPRM	Command and Parameters
<b>Related Message Groups</b>	
MGRPID1 to MGRPID5	Group Name
MCOR1 to MCOR5	Correlation Key

*(Sheet 3 of 4)*

Table G-4. *EventView Message Rule Field Names Used in the \$REDBAPI Procedure*

Field Names	Field Label on Panel
<b>Set Variables</b>	
SETVAR1 to SETVAR6 ( <i>name=value</i> ) or VAR1 and VALUE1 to VAR6 and VALUE6	Name and Value
<b>Rule Comments</b>	
COMMENT1 to COMMENT12	Comment Text
<i>(Sheet 4 of 4)</i>	

---

## \$REDBAPI SERVICE=DELETE

### Syntax

```
$REDBAPI    SERVICE=DELETE
            [NAME=rule-object-name]
            CLASS=cc
            RULESET=ruleset-name
```

### Use

Use this call to delete an EventView rule set or message rule definition from the knowledge base.

### Operands

#### **SERVICE=DELETE**

Indicates that a definition is to be deleted from the knowledge base.

#### **NAME=*rule-object-name***

Specifies the object name of the message rule definition to be deleted.

The name is contained in the &ZRENAME variable returned by a previous \$REDBAPI SERVICE=CREATE call.

This operand is not required if you are deleting a ruleset.

#### **CLASS=*cc***

Specifies the two digit identifier of the EventView class to which the definition belongs as follows: 93 for rulesets and 94 for message rules.

#### **RULESET=*ruleset-name***

Specifies the name of the ruleset to be deleted or the name of the ruleset that owns the message rule is to be deleted.

#### **Caution**

When you delete a ruleset, you also delete the rules it owns.

### Returned Variables

#### **&SYSMMSG**

Contains the message returned by \$REDBAPI.

## Return Codes

The following return codes indicate the success or failure of the deletion processing:

<b>&amp;RETCODE</b>	<b>Meaning</b>
0	Processing was successful.
8	Processing failed.
16	Error occurred in call syntax.

## Example

The following example deletes the the SUPP ruleset (class 93):

```
&CALL PROC=$REDBAPI +  
PARMS= ( SERVICE=DELETE , +  
CLASS=93 , +  
RULESET=SUPP )
```

---

## \$REDBAPI SERVICE=GET

### Syntax

```
$REDBAPI    SERVICE=GET  
            [NAME=rule-object-name]  
            CLASS=cc  
            RULESET=ruleset-name
```

### Use

Use this call to retrieve information about a EventView rule set or message rule definition in the knowledge base.

If you make this call from an NCL procedure, ensure that you share the &ZRE-prefixed variables, for example, by adding the following statement before the call:

```
&CONTROL SHRVAR=( ZRE )
```

### Operands

#### **SERVICE=GET**

Indicates that information about a definition is to be retrieved from the knowledge base.

#### **NAME=*rule-object-name***

Specifies the object name of the message rule definition for which information is to be retrieved.

The name is contained in the &ZRENAME variable returned by a previous \$REDBAPI SERVICE=CREATE call.

This operand is not required if you are retrieving information about a ruleset definition.

#### **CLASS=*cc***

Specifies the two digit identifier of the EventView class to which the definition belongs as follows: 93 for rulesets and 94 for message rules.

#### **RULESET=*ruleset-name***

Specifies the name of the ruleset for which information is to be retrieved or the name of the ruleset that owns the message rule for which information is to be retrieved.

## Returned Variables

### **&ZREfield-name**

Knowledge base information about the specified rule set or rule is returned in *&ZREfield-name* variables. For information about the field names, see the section, *EventView Definition Field Names*, on page G-55.

### **&SYSMMSG**

Contains the message returned by \$REDBAPI.

## Return Codes

The following return codes indicate the success or failure of the retrieval processing:

<b>&amp;RETCODE</b>	<b>Meaning</b>
0	Processing was successful.
8	Processing failed.
16	Error occurred in call syntax.

## Example

The following example retrieves information about a message rule (class 94) previously created in the SUPP ruleset:

```
&CALL PROC=$REDBAPI PARS=( SERVICE=CREATE , ... )

.
.
.
&JAA992I=&ZRENAME
.
.
.

&CALL PROC=$REDBAPI +
SHARE=( ZRE> ) +
PARMS=( SERVICE=GET , +
NAME=&JAA992I , +
CLASS=94 , +
RULESET=SUPP )
```

---

## \$REDBAPI SERVICE=LIST

### Syntax

\$REDBAPI	SERVICE=LIST CLASS= <i>cc</i> [RULESET= <i>ruleset-name</i> ] [RULEID= <i>message-rule-id</i> ]
-----------	--

### Use

Use this call to list EventView rule set or message rule definitions in the knowledge base.

If you make this call from an NCL procedure, ensure that you add the following statement before the call:

```
&CONTROL SHRVAR=( ZRELST )
```

### Operands

#### **SERVICE=LIST**

Indicates that selected definitions in the knowledge base are to be listed in &ZRELST*nmmn* variables.

#### **CLASS=*cc***

Specifies the two digit identifier of the EventView class to which the definitions belong as follows: 93 for rulesets and 94 for message rules.

#### **RULESET=*ruleset-name***

Identifies the rulesets to be listed, or specifies the name of the ruleset that owns the message rules to be listed.

If you are listing rulesets, the value is generic. For example, if RULESET=SUP, then rulesets with names that start with SUP are matched.

#### **RULEID=*message-rule-id***

Specifies the IDs of the message rules to list.

The ID is the first word of the message text.

The value is generic. For example, if RULEID=JBB1, then message rules with a first word that starts with JBB1 are matched.

## Returned Variables

### **&ZRELSTnnnn**

Each knowledge base definition entry is returned in an **&ZRELSTnnnn** variable.

### **&SYSMMSG**

Contains the message returned by \$REDBAPI.

## Return Codes

The following return codes indicate the success or failure of the list processing:

<b>&amp;RETCODE</b>	<b>Meaning</b>
0	Processing was successful.
8	Processing failed.
16	Error occurred in call syntax.

## Example

The following example retrieves the list of all rulesets:

```
&CALL PROC=$REDBAPI +  
SHARE=( ZRELST> ) +  
PARMS=( SERVICE=LIST , +  
CLASS=93 )
```

---

## SSI DD SUBSYS Support

This appendix describes the following:

**Note**

Support for DD SUBSYS is available on MSP and OS/390 systems only.

- The DD SUBSYS facility, its implementation and use
- Using the WTO facility to send messages to operator terminals
- Using PPI facilities to communicate with other processes
- How to write custom procedures for processing input and output
- The API for filter exits and user-written input/output procedures
- Sample exit procedures

---

## About DD SUBSYS

DD SUBSYS is a facility (provided by the system) that allows an authorized subsystem to provide access methods to existing programs. In effect, the subsystem appears to these programs as if it is a set of files, and in this way is able to provide data to, or receive data from, the programs.

For example, assume that you have a third-party program that writes an event log while it is running. The information in this event log could be very useful for system automation. Normally, however, the data written to this log cannot be accessed by another program until the program producing the log is shut down. DD SUBSYS allows you to intercept all data written to the file while logging is taking place. You can then use the WTO facility to send the data to operator terminals. The Advanced Operations Management (AOM) software then forwards the data to any specified NCL process in the region.

### Note

Regions that use Automation Services support only the WTO facility.

---

## Using DD SUBSYS

This section describes how to use DD SUBSYS in general terms. You may also want to refer to the *MVS/ESA JCL Reference* manual for further JCL considerations, and the *MVS/ESA Programming: Authorized Assembler Services Guide* for details of the Dynamic Allocation interfaces.

## Implementing DD SUBSYS

The DD SUBSYS facility can be implemented on your system as follows:

- Step 1. An extra operand on the JCL DD statement allows you to nominate the name of a subsystem that is to process the I/O requests of this file.

This is the SUBSYS operand and is used as follows:

```
//ddname DD SUBSYS=(parm1,parm2...parmn)
```

This operand tells the operating system that the nominated subsystem is to process this file.

### Note

If individual parameters of the SUBSYS operand contain special characters, including equals signs (=), then the subparameter must be enclosed in quotes. The quotes are removed before the parameter is passed to the subsystem.

Step 2. Equivalent facilities to the DD SUBSYS operand are available when using dynamic allocation. Two text units can be used, as follows:

```
DALSSNM X'005F'
```

Specifies the subsystem name (corresponding to the first subparameter of the SUBSYS operand in JCL).

```
DALSSPRM X'0060'
```

Specifies subsystem parameters (corresponding to the additional subparameters of the SUBSYS operand in JCL).

**Note**

Since each subparameter passed in the text unit has a length, the previous comments regarding quotes in the JCL parameter do not apply.

Step 3. The Subsystem Interface (SSI) provides several function codes that must be supported by a subsystem to allow it to use DD SUBSYS.

These function codes and their purpose are shown in the table below:

Function Code	Purpose
07	Unallocation (UN)—called at job step end or by dynamic deallocation of the file.
16	Open (OP)—called when the file is opened.
17	Close (CL)—called when the file is closed.
38	Converter/interpreter (CI) called when JCL is parsed (not used for dynamic allocation).
39	Allocation group (AG)—called at job step allocation or dynamic allocation time.

Step 4. The subsystem must also provide a set of routines that receive control whenever a program that has opened a DD SUBSYS file issues an I/O request to it. These routines get control as a logical subroutine called by the user program and must simulate the processing of normal I/O requests. How these routines provide data to the user program (for input) or data from the user program (for output) is up to you.

Step 5. DD SUBSYS interface routines in your operating system translates all DCB-based I/O requests to an ACB/RPL-based interface for the subsystem. This is the normal mode of operation. If the subsystem is coded with appropriate logic, it can handle an application program that opens an ACB to DD SUBSYS. In this case the subsystem could simulate VSAM.

## Who Can Use DD SUBSYS Facilities?

Any job in the same operating system as the providing subsystem can use the DD SUBSYS facilities. There is no requirement for the user of DD SUBSYS facilities to be in the same region as the subsystem itself. Of course, a specific implementation of DD SUBSYS may impose restrictions. The NMSSI implementation does *not* impose any restrictions.

Note, however, that an implementation of DD SUBSYS may not support some types of file or I/O. For example, the NMSSI implementation discussed here provides two functions (WTO and PPISEND) that are applicable to output files only. If you attempt to open these files for input the operation fails.

Similarly, some subsystems may not be able to simulate enough VSAM information for a program to open an ACB directly to the subsystem.

To use DD SUBSYS, the providing subsystem must be active, otherwise you get JCL errors, or ALLOCATION and OPEN statements fail.

---

## NMSSI DD SUBSYS Support

To use the DD SUBSYS support provided by NMSSI, you must first enable it as described in the following sections.

The NMSSI support for DD SUBSYS provides several facilities:

- The ability to generate WTO output records.
- The ability to send output records to a nominated PPI Receiver.
- The ability to act as a PPI receiver and to pass incoming records to a program as an input file.
- The ability to provide a general user function where a user-nominated program can be called to process I/O.

In addition, all the supplied facilities support two additional functions:

- The ability to copy the data records to another data definition.
- The ability to call a user-written filter program that can determine whether individual records are to be accepted or rejected.

These facilities are discussed in the following sections.

## Enabling DD SUBSYS Support in NMSSI

To enable the use of DD SUBSYS with NMSSI, use the following startup parameter:

```
DDSUBSYS={ NO | YES }
```

The following example enables SSI for DD SUBSYS:

```
//DDSUBSYS EXEC PGM=NMSSI,TIME=1440,  
//          PARM=( 'SSID=xxxx,DDSUBSYS=YES' )  
//SSIALOG  DD SYSOUT=*  
//SSIDUMP  DD SYSOUT=*  
//SYSUCUMP DD SYSOUT=*
```

The SSID used for this SSI needs to be specified in the IEFSSN<sub>xx</sub> or SUBSYS<sub>xx</sub> member in SYS1.PARMLIB.

You can start an SSI as a started task or as a job.

## Stopping the SSI

You can stop the SSI by using the following system operator command:

```
F job-name,SSI STOP
```

However, you should not stop and/or restart the SSI while using it to find DD SUBSYS—JCL errors, because allocation errors can occur.

Worse still, open errors can occur if the SSI is stopped after a job has started but before it opens a DD SUBSYS dataset. These errors cause OPEN ABENDS (013-C0) that can crash the user program.

Finally, if you stop the SSI while a DD SUBSYS dataset is open, the next I/O request to that dataset will ABEND with a U0001 ABEND code. This will be preceded by a message that is sent as a WTO message to the system console:

```
NS4199 ACCESS ERROR on ddname
```

Restarting the SSI does not prevent ABENDs or errors from occurring in jobs that are active but dormant. Jobs that require the SSI that has been stopped (and restarted) for DD SUBSYS do *not* automatically use the restarted SSI. Subsequent SSI requests from those jobs can fail because the original SSI has stopped. Restart those jobs to use the new SSI.

Most of these restrictions are due to operating system limitations.

## General NMSSI DD SUBSYS Syntax

All supplied functions of the NMSSI DD SUBSYS facility use a common syntax:

```
//ddname DD SUBSYS=(ssid,function[, 'PARM=value',...])
```

The operands are as follows:

### *ssid*

The NMSSI SSID.

### *function*

A supported function name (see the next section, *Supported Function Names*). This *must* be the second subparameter.

### **'PARM=value'**

One or more functions or general parameters. Each parameter is a keyword/value pair. Due to JCL requirements, you *must* use quote marks, because there are special characters to specify, such as the equals sign (=). The parameter names depend on the specific function.

When using dynamic allocation, the subparameters are provided as individual length/value pairs on the DALSSPRM text unit. In this case, there is no need to quote the parameter values.

## Supported Function Names

The following function names are supported:

### **WTO**

Invokes the WTO facility that allows output records to be sent to the system console through the WTO macro.

### **PPISEND**

Invokes the PPI SEND function that sends output records through PPI to any PPI receiver.

### **PPIRECV**

Invokes the PPI RECEIVE function that allows records sent to the nominated PPI receiver name to be fed to a program as input.

### **USER or USERI or USERO**

Invokes the USER facilities that allow a user-specified program to be driven to process input or output records.

Full details on each function and its relevant subparameters are contained in the following sections.

## Common Functions

As mentioned above, there are two additional facilities available when using NMSSI DD SUBSYS functions. These facilities enable the copying of data to another file and the filtering of data through a user exit program.

### COPY Facility

The COPY facility allows you to copy the DD SUBSYS file records (input or output) to another file.

This allows you to see and process a data flow while using DD SUBSYS, while still sending the data to its original destination. For example:

```
//LOG DD SUBSYS=(NMSS,WTO,'COPY=LOG2')
//LOG2 DD SYSOUT=A
```

The format of the COPY operand is:

```
'COPY=ddname'
```

*ddname* is the target data definition name. There are special DCB attribute defaulting rules. You can override these by using a DCB parameter on the target COPY *ddname*.

The rules are as follows:

- If the DCB has no RECFM specified, then the attributes are forced to RECFM=U, LRECL=0, and BLKSIZE=6233; however:
  - If RECFM is U and BLKSIZE is specified, then the specified BLKSIZE is used.
  - If RECFM is U and no BLKSIZE is specified, then BLKSIZE=6233 is set.
- If LRECL=0, and:
  - If the RECFM is FB, then LRECL=132 is set, or LRECL=133 if the RECFM has A or M. If B (Blocked), LRECL\*40 is set as the BLKSIZE, otherwise the BLKSIZE is set to the LRECL.
  - If the RECFM is VB, then LRECL=136 is set, or LRECL=137 if the RECFM has A or M. If B (Blocked), 6233 is set as the BLKSIZE, else the BLKSIZE is set to the LRECL+4.
- If BLKSIZE=0, and:
  - If the RECFM is U, BLKSIZE=6233 is set.
  - If the RECFM is F, BLKSIZE=LRECL is set.
  - If the RECFM is FB, a BLKSIZE that is the largest integral multiple of the LRECL less than 6233 is set. Otherwise, the value of the LRECL is used, if it is greater than 6233.
  - If the RECFM is V, BLKSIZE=LRECL+4 is set.
  - If the RECFM is VB, BLKSIZE=6233 is set (if LRECL is less than 3110), otherwise BLKSIZE=LRECL+4 is set.

Note that you can cascade DD SUBSYS statements; the DD statement for the ddname specified for one COPY operand can itself be a DD SUBSYS definition. This can allow several facilities to see output from a program.

**Note**

The order in which the FILTER and COPY statements are used on a single DD SUBSYS statement is important. If the FILTER operand precedes the COPY operand, then the COPY facility only copies records that were not rejected by the filter exit. If the FILTER operand follows the COPY operand, then all records are copied, regardless of the result of the filtering process. If the COPY file has either an I/O error or an ABEND, then the application program abends.

## FILTER Facility

The FILTER facility provides for a user exit program to be called to determine whether or not each individual record is to be passed on for processing.

If processing an output file, for example, for PPISEND, you can use the FILTER facility to prevent a record from being sent.

If processing an input file, you can use the FILTER facility to prevent a record from being passed to the user program.

Additionally, if the COPY facility (see the previous section on page H-7) is also being used on the same DD statement, filtering may affect the copy facility.

The syntax of the FILTER operand on DD SUBSYS is:

```
'FILTER=pgmname'  
'FILTER=(pgmname)'  
'FILTER=(pgmname, )'  
'FILTER=(pgmname,parm)'
```

Due to JCL requirements, you *must* use quote marks when specifying the FILTER operand, because it contains special characters. (Quoting is not necessary, however, if you are using dynamic allocation.)

The FILTER operand can intermingle with other DD SUBSYS subparameters, although there is an interaction between it and the COPY parameter (see the Note: below).

The *pgmname* operand specifies the name of a user exit program. This program is loaded during OPEN processing for the file. It is then called to open the file, once for each GET or PUT of a logical record, and to close the file.

The *parm* operand is optional. If specified, it must be a one-character to eight-character value, in PDSNAME format. It is passed to the user filter exit. The exit can use it as, for example, a data definition name (ddname) of a control file.

Details of the filter program API are discussed in the section, *FILTER Exit API*, on page H-18 of this appendix. UTIL0037, a sample filter program that is supplied, is described in the section, *Sample FILTER Exit—UTIL0037*, on page H-33 of this appendix.

The filter program need not be APF authorized, unless the application program is APF authorized. In this case, the filter program *must* come from an APF-authorized library.

**Note**

If using both the FILTER and COPY facilities on a single DD SUBSYS statement, the order is important. If the FILTER operand precedes the COPY operand, then the COPY facility only copies records that were not rejected by the filter exit. If the FILTER operand follows the COPY operand then all records are copied regardless of the result of the filtering process.

If the FILTER program abends while processing a FILTER request, the application program also abends.

## Carriage Control

When an output dataset has carriage control specified (that is, the RECFM has A or M in it), the following special considerations apply:

- The ACB/RPL interface does not normally provide the control character as part of the record. Rather, the control character is pointed to separately. Because of this, NMSSI DD SUBSYS support normalizes the record.
- When using facilities such as PPISEND or WTO, you probably do not want to send the control character. These facilities default to not sending it, but this can be overridden.
- The COPY facility always copies the entire record, including the control character.
- The FILTER facility user exit is passed the entire record, including the control character, and is notified of its existence so that it can take appropriate processing steps.

---

## WTO Facility

The WTO facility provided by NMSSI allows you to send records that are written to a file as WTO messages. This allows the messages to be viewed by an operator, and/or picked up by an automation product (for example, SOLVE:Operations Automation).

**Note**

Use the WTO facility sparingly, otherwise you can fill up console buffers and crash the system.

A logical record can have an optional prefix added to the front, then the total record is truncated to 126 characters before being sent as a WTO message.

## WTO DD SUBSYS Operands

The following are the DD SUBSYS operands for the WTO function. Note that other than for the WTO name itself, the order of the operands is not important:

### **WTO**

Specifies the WTO function and *must* be the first subparameter after the subsystem name.

### **‘ROUTCDE=(list)’**

Specifies an optional list of routing codes to apply to the WTO messages. If this operand is omitted, ROUTCDE=11 is assumed. Note that quote marks are required. The list of routing codes can be a single number (in which case, no parentheses are necessary) or a list of numbers from 1 through 16 (MSP) or 1 through 128 (OS/390).

#### **Note**

If using routing codes greater than 20, authorization is required. Unauthorized programs cannot issue WTO messages using these routing codes and, if they do, an ABEND D23 results.

### **‘DESC=(list)’**

Is an optional list of descriptor codes to apply to the WTO messages. If this operand is omitted, DESC=7 is assumed. Syntax is as for ROUTCDE, values range from 1 through 16.

#### **Note**

It is not advisable to use descriptor codes 1, 2, or 11, as these messages will stay on the consoles.

### **‘PREFIX=value’**

Allows for the specification of an optional WTO prefix. If this operand is omitted, the SSID is used (but you can force the prefix to be dropped by coding PREFIX=NO). Otherwise, you can specify a 1-character to 12-character prefix value. There must be a single blank between the prefix and the text of the message.

### **‘BASE=n’**

Control the starting column number for the data that is sent as a WTO message. The value of *n* must be a number from 1 through 32760. It is a logical column number. The default value is 1 if the file does not have carriage control; otherwise, it defaults to 2. This means that a print file will not normally have carriage control characters in the WTO text. BASE can also be used to skip a record prefix, and so on.

### **‘COPY=ddname’**

### **‘FILTER=parms’**

These parameters allow for copying and filtering the data. See page H-7 and page H-9, respectively.

## Notes

You need to consider the following:

- A null record after BASE considerations is ignored.
- Routing codes above 16 are not supported unless the operating system is OS/390.
- The FILTER facility is especially useful here to limit the amount of data sent to the consoles.

---

## PPISEND Facility

The PPISEND facility provided by NMSSI allows you to send logical records written to a file to any PPI receiver. The receiver can be, for example, an NCL process in a region. It can also be any user program that uses the PPI API.

## PPISEND DD SUBSYS Operands

The following are the DD SUBSYS operands for the PPISEND function. Note that other than for the PPISEND name itself, the order of the operands is not important:

### **PPISEND**

Specifies the PPISEND function and *must* be the first subparameter after the subsystem name.

### **'TARGET=*targetid*'**

Nominates the identifier of the PPI receiver. *targetid* must be a valid PPI receiver name. The name must be defined to PPI at the time the dataset is opened (although the PPI need not be active). This operand is required.

**‘SOURCE=*sourceid*’**

Provides an optional PPI sender ID. This must be a valid PPI name, but is not registered to PPI. As well as a valid name, the following special names are supported:

**\*JOB**

Use the job name (this is the default).

**\*STEP**

Use the job step name. If the job step name does *not* exist, use the job name.

**\*PSTEP**

Use the procedure step name. If the procedure step name does *not* exist, use the step name. If the step name does *not* exist, use the job name.

**‘QFULL=*option*’**

Control the action to take if the PPISEND receives a PPI *queue full* condition (PPI return code 35). The default is ERROR. The following actions can be specified:

**ERROR**

Specifies that a *Dataset Full* condition be reflected to the application program with an RPL error of 28 (X‘1C’). If the application is using a DCB (as is normally the case), then this is reflected as an I/O error (not an X37 ABEND).

**IGNORE**

Specifies that the queue full return code be ignored. This means that the record is not sent; however, processing continues. Note that, if you are using the COPY facility, the COPY is still performed.

**‘BASE=*n*’**

Controls the starting column number for the data that is sent to PPI. *n* must be a number from 1 through 32760. It is a logical column number. The default value is 1, if the file does not have carriage control; otherwise, it defaults to 2. This means that a print file will not normally have carriage control characters sent across PPI. BASE can also be used to skip a record prefix, and so on.

**‘COPY=*ddname*’**

**‘FILTER=*parms*’**

These parameters allow for copying and filtering of data. They are documented on page H-7 and page H-9, respectively.

You need to consider the following:

- The *targetid* and *sourceid* values are validated at JCL syntax check time. If they are invalid then a JCL error occurs. This syntax check is repeated at step allocation time.
- When the dataset is opened, PPI is called to check that the receiver exists. If PPI is inactive, or if the receiver is not defined, then the open fails. This leads to an 013-C0 OPEN ABEND occurring for a DCB. For an ACB, an ACB open error code is set. Messages are written informing of the problem. Any other unexpected PPI return codes also cause an open error.
- The dataset must be open for output, otherwise the open fails.
- When records are written to the dataset, a PPI SEND is issued for each logical record. Any return code other than 0 (All OK), or 4 (Inactive But Queued) results in an I/O error. The return code of 35 (Receiver Queue Full) is translated into an RPL error code of 28 (X'1C')—Dataset Full, for ACB/RPL-based callers (if QFULL=ERROR is in effect; otherwise, it is ignored).
- The PPI receiver receives the logical records (with no prefix, even if the user program is writing V or VB), exactly as sent, or as adjusted by the BASE value. The receiver can use the sender ID to determine where the records came from.
- No null record (not even one that is logically null after considering the BASE value) is sent.

---

## PPIRECV Facility

The PPIRECV facility provided by NMSSI allows you to receive data from other PPI senders and to provide the received data as an input file. The senders can be any other PPI users, including NCL processes in a region. They can also be any program that uses the PPI API.

### PPIRECV DD SUBSYS Operands

The following DD SUBSYS operands are required or optional with the PPIRECV function. Note that other than for the PPIRECV name itself, the order of the operands is not important:

#### **PPIRECV**

Specifies the PPIRECV function and *must* be the first subparameter after the subsystem name.

**‘ID=*name*’**

Nominates the PPI receiver name. *name* must be a valid PPI receiver name. The name must *not* be defined to PPI at the time the dataset is opened or, if defined, must *not* be active. This operand is required.

**‘APF=*option*’**

Allows you to specify whether or not PPI senders must be APF authorized to send to this name.

**NO**

Specifies that senders do not need APF authorization. This is the default.

**YES**

Specifies that senders must be APF authorized.

**‘IQEMPTY=*option*’**

**‘QEMPTY=*option*’**

Specifies the action to take when a PPI Queue Empty return code (30) is returned. The two operands allow a different action to be taken for the initial read request (IQEMPTY), and for all subsequent read requests (QEMPTY).

The following actions can be specified:

**EOF**

Specifies that an end-of-file is returned to the application. This is the default for QEMPTY.

**WAIT**

**(WAIT)**

Specifies that an indefinite wait is performed until data is queued. This is the default for IQEMPTY.

**(WAIT,*n*)**

Specifies that a wait for an interval of *n* seconds is performed. *n* is in the range 1 through 86400. If no data arrives in this interval, an end-of-file is returned.

**‘MAXQUEUE=*n*’**

Specifies the PPI receiver buffer queue limit. The value of *n* must be within the range 1 through 9999. The default is 10.

**'COPY=ddname'**

**'FILTER=parms'**

These parameters allow for copying and filtering the data. See page H-7 and page H-9, respectively.

**Note**

Since PPIRECV is an input facility, the filtering is performed on records received from PPI, before being returned to the application as input records.

## Notes

You need to consider the following:

- If the program opens the file as RECFM=F or FB, short input records will be padded with blanks to the LRECL.
- Records that are too long will cause an I/O error.
- Use of IQEMPTY and QEMPTY can allow you to set up a job to wait indefinitely for initial input, then process input until no more arrives within a certain time, then return EOF and terminate.
- If you are filtering, you might not get a return on input for several wait intervals. This is because, if the filter program rejects the input record, the wait is re-executed.

---

## USER, USERI, and USERO Facilities

The USER $x$  facilities allow you to supply your own DD SUBSYS I/O routines. This is an open-ended facility. If one of the supplied functions does not allow you to do what you want to do, then you can use the USER $x$  facilities to implement it.

Full details on the API for the user program are provided as well as a sample. See *Notes* at the end of this section for further details.

## USERx DD SUBSYS Operands

The following DD SUBSYS operands are used with the USERx function. Note that other than for the USERx name itself, the order of the operands is not significant.

### USER

### USERI

### USERO

Specifies the user function to be performed and must be the first subparameter after the subsystem name.

USER allows the user program to handle most dataset options. The only option that is blocked by the NMSSI front-end is the use of ASY mode I/O.

USERI allows the user program to handle sequential input only. This is the most common type of input processing and removes the need for the user program to validate ACB and RPL options.

USERO allows the user program to handle sequential output only. This is the most common type of output processing and removes the need for the user program to validate ACB and RPL options.

### **'PGM=*pgmname*'**

Names the user program. *pgmname* must be a valid program name. The program must be available for loading at OPEN time (for example, in the STEPLIB or link list).

The user program need not be APF authorized, unless the application opening the file is APF authorized, in which case the user program *must* come from an APF-authorized library. The PGM operand is required.

### **'PARAM=*parm*'**

Provides an optional one-character to eight-character parameter to be passed to the user program. If specified, the value must be in a valid partitioned dataset (PDS) name format. If omitted, an 8-character value of all blanks is used.

### **'COPY=*ddname*'**

### **'FILTER=*parms*'**

These parameters allow for copying and filtering the data. See page H-7 and page H-9, respectively.

If the file is being written to, filtering and copying occur before the user program is called. Records rejected by filtering are not passed to the USER program.

If the file is being read, filtering and copying is performed on the records provided by the USER program. If the filter program rejects a record, the user program is called to provide another one.

## Notes

The user program API is documented in the section, *USERx Facility API*, on page H-24 in this appendix.

A sample user program is provided in source form (UTIL0038). Details of this program are provided in the section, *Sample USER Exit—UTIL0038*, on page H-37 in this appendix.

---

## **FILTER Exit API**

This section describes the supplied API that allows you to write a filtering program.

The filter program can be used to select which records that are written to or read from a DD SUBSYS facility are actually processed (written records), or returned to the requester (records read).

The filter program can be written in any language that supports standard linkage conventions. However, for performance reasons, it is probably best written in assembler.

The sample filter program that is supplied contains extensive comments.

## APF Authorization

The use of DD SUBSYS could provide an opportunity to circumvent system security. This is because the exit programs run during OPEN and CLOSE processing, and normally the system is in supervisor state or has a protection key set (to less than 8) at these times.

Consequently, the following rules are implemented:

- If the application program (that is, the job step program) is not APF authorized, then the filter exit need not be APF authorized, and need not reside in an APF-authorized library. Although it can be APF authorized, and can reside in an APF-authorized library, the APF authorization is ignored.
- If the application program is APF authorized, then the filter exit must come from an APF-authorized library, although the exit need not be APF authorized itself. Since APF libraries are normally security protected, this prevents a potentially dangerous filter program from being loaded and executed in an environment where APF authorization exists.

### Note

Specifying any unauthorized step libraries results in all step libraries being considered not authorized for that step.

- The OPEN and CLOSE calls to the filter exit are made in the job step TCB key and state.
- The FILTER calls to the filter exit are made directly from the application, thus the system state and key at this time is dependent on the application.

The above rules mean that a simple application can use a filter program without any need to access APF libraries, and so on. Only when using authorized applications is there any need to place the filter program in an APF library.

## Abnormal Termination (ABEND)

If the filter program abends during an OPEN or CLOSE call, the ABEND is trapped by the DD SUBSYS code and is reflected as an OPEN error (during an OPEN), or is ignored (during a CLOSE).

If the filter program abends during a FILTER call, the ABEND is reflected by the application (as the filter program is merely being called as a subroutine of the application program). For this reason, you should ensure that the filter program is well tested before placing it in a production environment.

## Dynamic Allocation

While the user program can open its own files at any time (including during an OPEN and a CLOSE call), it cannot make Dynamic Allocation requests during OPEN or CLOSE calls. This is because a system ENQ is held, and a call to Dynamic Allocation results in an error.

Thus, the user program should have any required files pre-allocated in the JCL before an OPEN is performed on the DD SUBSYS file.

## Calling Details

The filter program is called with registers set up as follows:

<b>Register</b>	<b>Content</b>
R0	Indeterminate.
R1	Points to a parameter list, as described below.
R2...R12	Indeterminate.
R13	Points to a standard 18-word save area.
R14	Contains the return address and AMODE.
R15	Contains the entry point and filter exit AMODE.

In a 31-bit environment, the filter exit is called in the AMODE, as established by the linkage editor. It need not return in the AMODE specified in register 14 on entry (that is, it can use a BR R14 instruction to return), because the return address is guaranteed to be below the line. The caller of the filter exit restores its own AMODE.

For the OPEN and CLOSE calls, the PSW key and state is as for the job step program (normally key 8, problem state).

For the FILTER calls, the PSW key and state is as for the application program at the time it issued the I/O request. This is normally key 8, problem state.

On completion, the filter program must restore registers 2 through 12 to the values they had when the filter program started processing.

## Return Codes

The filter exit sets a return code in register 15 to reflect the results of its processing.

Setting a non-zero value for the OPEN call means that the attempted opening of the dataset failed. An optional error message can be supplied in this case.

The return code for the CLOSE call is ignored.

A non-zero return code for the FILTER call means that the current record is to be rejected.

## Reentrancy

The filter exit need not be written to be reentrant. However, if the same filter exit is to be used for more than one DD SUBSYS filter in a single job step, it is a good idea to make it reentrant. This is because each OPEN will cause a separate non-reentrant copy to be loaded. However, at CLOSE time, there is no way to tell the system which copy is to be deleted. This means that an active copy of the exit could be deleted instead of the one that has been closed.

## Parameter List

The parameter list for the filter exit is provided by the Assembler macro \$NMDDSF. The individual fields in the parameter list are described below.

Remember that the parameter list is pointed to by register 1 on entry to the filter exit. Many of the parameters are in storage and cannot be altered by the filter exit.

In the following description, the name of the pointer in the parameter list is provided. The target field is described.

The fields in the parameter list are as follows:

### **FPLS@FC**

A pointer to a binary full word that contains a function code. The function code determines what processing is required. Valid function code values are:

- 0—OPEN CALL
- 4—CLOSE CALL
- 8—FILTER CALL

### **FPLS@DDN**

A pointer to the eight-character, blank padded ddname for the file being filtered. This ddname is in protected storage and cannot be altered.

### **FPLS@PRM**

A pointer to the eight-character, blank padded parameter value specified as the filter (for example: 'FILTER=(MYPGM,MYPARM)' would result in the parameter being CL8'MYPARM'). If no value is specified, then the field is blank.

This parameter value is in protected storage and cannot be altered.

#### **Note**

If a value is specified, it is edited to be a valid ddname (or PDS member name). This means that no editing is required if you wish to use it as a ddname or member name (although obviously you need to test for the existence of the ddname or member name).

### **FPLS@JFC**

A pointer to the Job File Control Block (JFCB) for the file. This control block contains various useful fields, such as the DSNNAME (assuming one was specified along with the SUBSYS parameter; otherwise a system-generated name is provided).

The IEFJFCBN mapping macro maps this control block. The JFCB is in protected storage.

### **FPLS@UWD**

A pointer to a four-byte, aligned area, initialized to binary zeros prior to the OPEN call. You can update this value. The updated value is then passed to subsequent FILTER calls and to the CLOSE call. If the value is further updated, then the new value is passed on subsequent calls.

An excellent use for this field is to anchor a work/save area that you obtain on the OPEN call. This makes it very easy to make the filter program fully reentrant.

#### **Note**

If a non-zero return code results from the OPEN call (that is, the OPEN fails), then the program is not called again. In this case, any work areas obtained should also be freed, as the CLOSE call is unable to free them later.

**FPLS@FLG**

A pointer to a one-byte flag, in protected storage. This flag contains several useful equated bits (the equates are in \$NMDDSPF):

**FFLG1OIN X'80'**

The file is open for input.

**FFLG1OOT X'40'**

The file is open for output (FFLG1OIN can also be set).

**FFLG1OUP X'20'**

The file is open for update (both FFLG1OIN and FFLG1OOT are set).

**FFLG1CCH X'01'**

The file is open for output and there are control characters present (that is, the original dataset had A or M in the RECFM).

**FPLS@LEN**

A pointer to a length field.

For the OPEN call, it points to a full word in protected storage that has the value F'120'—this is the length of the supplied error message return area.

For the CLOSE call, it points to a full word in protected storage that has the value F'0'.

For the FILTER call, it points to a full word that contains the length of the record that is to be filtered. Note that this value is not in protected storage, but that altering it will not have any effect on other processing.

**FPLS@REC**

A pointer to the record area.

For the OPEN call, it points to a 120-byte, blanked area. This area can be used to return an error message when you return a non-zero return code to fail the open. In this case, if the area is not blank, the message is sent through WTO to inform of the open failure reason.

For the CLOSE call, it points to a full word F'0' in protected storage.

For the FILTER call, it points to the record to be filtered. This record must not be altered. There is no guarantee that you will be able to access storage past the length pointed to by FPLS@LEN.

The record starts with the control character, if one is present.

**FPLS@ACB**

A pointer to the ACB for the file. On OPEN, this ACB is a copy in protected storage. On CLOSE and FILTER, this is the real ACB. It must not be altered.

**FPLS@RPL**

A pointer to the RPL for the I/O request. For OPEN and CLOSE, it points to a dummy full word 0 in protected storage. For I/O, it points to the real RPL. Because the record address, length, handle control character prefixing, and LOCATE mode are provided, there should be little need to actually refer to the RPL.

The parameter list provides all the information that is needed to perform record filtering.

## Sample Filter Exit

The section, *Sample FILTER Exit—UTIL0037*, on page H-33 of this appendix describes a sample filter exit, UTIL0037, that is supplied in source form. It illustrates the use of the filter parameters to provide a general-purpose filter facility.

---

## USERx Facility API

This section describes the API that enables you to write your own DD SUBSYS I/O handler.

The user program can be used to replace any DD SUBSYS function with your own code. Some suggestions are:

- Reformatting files for programs that cannot be rewritten
- Encryption and decryption of data without intermediate files
- Real-time monitoring of messages

The user program can be written in any language that supports standard linkage conventions. However, for performance reasons, it is best written in assembler.

The user program includes extensive comments.

## APF Authorization

The use of DD SUBSYS could provide an opportunity to circumvent system security. This is because the user programs run during OPEN and CLOSE processing, and normally the system is in supervisor state or has a protection key set (to less than 8) at these times.

To avoid security exposure, the following rules are implemented:

- If the application program (that is, the job step program) is not APF authorized, then the user program need not be APF authorized, and need not reside in an APF-authorized library. Although it can be APF authorized, and can reside in an APF-authorized library, the APF authorization is ignored.
- If the application program is APF authorized, then the user program must come from an APF authorized library, although the program need not be APF-authorized itself. Since APF libraries are normally security protected, this prevents a dangerous user program from being loaded and executed in an environment where APF authorization exists.

### Note

Specifying any unauthorized step libraries results in all step libraries being considered not authorized for that step.

- The OPEN and CLOSE calls to the user program are made in the job step TCB key and state.
- The I/O calls to the user program are made directly from the application, thus the system state and key at the time is dependent on the application.

The above rules mean that a simple application can use a user program without any need to access APF libraries, and so on. Only when using authorized applications is there any need to place the filter program in an APF library.

## Abnormal Termination (ABEND)

If the user program abends during an OPEN or CLOSE call, the ABEND is trapped by the DD SUBSYS code and reflected as an OPEN error (during an OPEN call), or is ignored (during a CLOSE call).

If the user program abends during an I/O call, the ABEND is reflected by the application (as the user program is merely being called as a subroutine of the application program). For this reason, ensure that the user program is well tested before placing it in a production environment.

## Supported I/O Requests

The USER<sub>x</sub> facilities allow most I/O requests to be processed. The restrictions are as follows:

- For the USER option, any Open mode is allowed. The only I/O option that is not allowed is ASY (asynchronous).
- For the USERI option, Open For Input Only is allowed. The only I/O request that is permitted is for synchronous input.
- For the USERO option, Open For Output is required. The only I/O request that is permitted is for synchronous output.
- The DD SUBSYS code always checks and disallows illogical conditions such as PUTLOCATE (not allowed by VSAM).
- The API hides most details of the ACB/RPL interface. Only the USER option needs to examine the ACB and RPL. USERI and USERO are protected from most things, including the use of GETLOCATE.

## Dynamic Allocation

While the user program can open its own files, and so on, at any time (including during the OPEN and CLOSE calls), it cannot make dynamic allocation requests during OPEN or CLOSE calls. This is because a system ENQ is held and calling dynamic allocation results in an error.

Thus, the user program should have any required files pre-allocated in the JCL before an OPEN is performed on the DD SUBSYS file.

## Calling Details

The user program is called with registers set up as follows:

<b>Register</b>	<b>Content</b>
R0	Indeterminate.
R1	Points to a parameter list, as described below.
R2...R12	Indeterminate.
R13	Points to a standard 18-word save area.
R14	Contains the return address and AMODE.
R15	Contains the entry point and filter exit AMODE.

In a 31-bit environment, the user program is called in the AMODE, as established by the linkage editor. It need not return in the AMODE specified in R14 on entry (that is, it can just use a BR R14 instruction to return), because the return address is guaranteed to be below the line. The caller of the user program restores its own AMODE.

For the OPEN and CLOSE calls, the PSW key and state is as for the job step program (normally key 8, problem state).

For the I/O calls, the PSW key and state is as for the application program at the time it issued the I/O request. This is normally key 8, problem state.

On completion, the user program must restore registers 2 through 12 to the values they had when the user program started processing.

## Return Codes

The user program reflects its processing by setting a return code in register 15.

Setting a non-zero value for the OPEN call means that the OPEN of the dataset failed. An optional error message can be supplied in this case.

The return code is ignored for the CLOSE call.

A non-zero return code for the I/O call causes an RPL logical error code to be set, unless the user program has already set a return error code. This means that you should normally return R15 = 0.

## Reentrancy

The user program need not be written to be reentrant. However, if the same user program is to be used for more than one DD SUBSYS file in a single job step, it is a very good idea to make it reentrant. This is because each OPEN causes a separate, non-reentrant copy to be loaded. However, at CLOSE time, there is no way to tell the system which copy is to be deleted. This means that a still active copy of the program could be deleted, instead of the one that has been closed.

## Parameter List

The parameter list provided to the filter exit is included in the Assembler macro \$NMDDSUP. The individual fields in the parameter list are described below.

Remember that the parameter list is pointed to by R1 on entry to the user program.

Many of the parameters are in protected storage and cannot be altered by the user program.

In the following description, the name of the pointer in the parameter list is provided. The target field is described.

The fields in the parameter list are:

### **UPLS@FC**

A pointer to a binary full word that contains a function code. The function code determines what processing is required. The function code values are:

- 0—OPEN call
- 4—CLOSE call
- 8—I/O call (see UPLS@IOF).

### **UPLS@DDN**

A pointer to the eight-character, blank-padded ddname for the file being processed. This ddname is in protected storage and cannot be altered.

### **UPLS@PRM**

A pointer to the eight-character, blank-padded parameter value supplied as the value for the PARM operand (for example, 'PGM=MYPGM', 'PARM=MYPARM', would result in the parameter being CL8'MYPARM '). If no value is supplied, then the field is blank.

#### **Note**

If a value is supplied, it is edited to be a valid ddname (or PDS member name). This means that no editing is required if you wish to use it as a ddname or member name (although obviously you need to test for ddname or member name existence).

This parameter is in protected storage and cannot be altered.

### **UPLS@NME**

A pointer to the eight-character, blank-padded USER<sub>x</sub> name. This will have the value USER, USERI, or USERO. By checking this value on OPEN, you can ensure that the I/O calls will only be made for expected values (for example, if the value is USERI, we guarantee that only synchronous GET requests are allowed).

If you allow the value USER, you need to be prepared to handle any I/O request type (particularly if the application opens an ACB as if it were a VSAM file).

### **UPLS@UWD**

A pointer to a four-byte, aligned area, initialized to binary zeros prior to the OPEN call. You can update this value. The updated value is then supplied to subsequent I/O calls and to the CLOSE call. If the value is further updated, then the new value is supplied on subsequent calls.

An excellent use for this field is to anchor a work/save area that you obtain on the OPEN call. This makes it very easy to make the USER program fully reentrant.

#### **Note**

If a non-zero return code results from the OPEN call (that is, the OPEN fails), then the program is not called again. In this case, any work areas obtained should also be freed, as the CLOSE call is unable to free them later.

### **UPLS@FLG**

A pointer to a one-byte flag that is in protected storage. This flag contains several useful equated bits (the equates are in \$NMDDSUP):

#### **UFLG1OIN X'80'**

The file is open for INPUT.

#### **UFLG1OOT X'40'**

The file is open for OUTPUT (UFLG1OIN can also be set).

#### **UFLG1OUP X'20'**

The file is open for UPDATE (both UFLG1OIN and UFLG1OOT are set).

#### **UFLG1FIX x'02'**

The dataset records are fixed length.

#### **UFLG1CCH X'01'**

The file is open for output and there are control characters present (that is, the original dataset had A or M in the RECFM).

### **UPLS@RLN**

A pointer to a binary half word (two bytes long) that contains the maximum record length. This is the record length as defined on OPEN and is data only. When UFLG1FIX is on, this value is useful for determining the correct record length to return for GET requests to prevent I/O errors.

This field is in protected storage and cannot be altered.

### **UPLS@IOF**

A pointer to the I/O function code for I/O calls only (points to a dummy full word 0 for OPEN/CLOSE). The values of this function code are re-equated in the IFGRPL Assembler macro for the RPLREQ field. The field is a binary full word.

To avoid the need for USERI and USERO programs to refer to the IFGRPL macro, the GET and PUT equates are defined in the \$NMDDSUP macro:

```
RPLGET      X'00000000'    (UPLSFGET)
RPLPUT      X'00000001'    (UPLSFPUT)
```

### **UPLS@IOA**

A pointer to an I/O area.

For OPEN, points to a 120-byte blank-padded error message area. You can set an error message here for output if you fail the OPEN call.

For CLOSE, points to a full word 0 in protected storage (cannot be written to).

For I/O (GET/PUT requests only for the USER program), points to an I/O area. This might or might not be the actual user I/O area, but this is not significant. A work I/O area is supplied in the case of GET LOCATE.

For other I/O requests (other than GET/PUT), it points to a full word 0 in protected storage.

## **UPLS@IOL**

A pointer to the length field for I/O.

For OPEN, points to a full word with the value 120 (the length of the error message area) in protected storage.

For CLOSE, points to a full word 0 in protected storage.

For PUT I/O requests, points to a full word that contains the length of the record being output.

For GET I/O requests, points to a full word containing the length of the I/O area. You must update this full word to be the actual length of the record that you are providing. Note that you do not need to make any special considerations for GET LOCATE, as a dummy record area is supplied in this case.

For other I/O requests (USER only), points to a full word 0 in protected storage.

## **UPLS@ERF**

A pointer to a two-byte error return field for I/O procedures.

For OPEN and CLOSE, points to a full word 0 in protected storage.

For I/O, points to a two-byte field that is initialized to binary zeros. This is where you can return RPL error codes as required. If no errors are to be returned, then these fields can be ignored.

The first byte must be set to X'08' for a logical error, or X'0C' for a physical error.

The second byte must be set to the reason code. For example, if you are returning a logical error (X'08'), you could set X'04' for EOF (input) or X'1C' for dataset full (output).

If you set a non-zero value in the first byte other than 08 or 0C, or set a non-zero value in the second byte and leave byte 1 zero, then the interface routines force an error code of X'08' with reason code X'DA'.

### **Note**

If you want the RPL error fields to be set directly, you leave these fields set to zero.

**UPLS@DSN**

A pointer to the 44-character dataset name (dsname).

This is the dsname specified in the DD SUBSYS statement, or a system-generated name if you did not specify a dsname. This field is in protected storage and cannot be altered.

**UPLS@ACB**

A pointer to the ACB for the file.

For OPEN, this ACB is a copy in protected storage.

For CLOSE and I/O, this is the real ACB. It must not be altered.

Note that if the application program opened a DCB, this ACB is the dummy one built by the SAMSII routines.

If you need to map the ACB, the appropriate Assembler macro is IFGACB.

**UPLS@DEB**

A pointer to the DEB for the file.

This is a dummy DEB as built for DD SUBSYS files. It is in protected storage and cannot be altered. The IEZDEB Assembler macro maps this control block.

**UPLS@JFC**

A pointer to the Job File Control Block (JFCB) for the file. This control block contains various useful fields, such as DSNAME (assuming one was specified along with the SUBSYS parameter; otherwise, a system-generated name is supplied).

The IEFJFCBN Assembler mapping macro maps this control block. The JFCB is in protected storage.

**UPLS@SOB**

For OPEN and CLOSE, points to the SSOB for the OPEN (SSOBFUNC=16) and CLOSE (SSOBFUNC=17) calls.

The SSOB can be used to locate other SSI control blocks. These control blocks are all in protected storage. The IEFJSSOB DA Assembler macro can be used to obtain the SSOB and OPEN/CLOSE extension maps.

For I/O, the pointer is a zero (that is, it points to nothing).

## **UPLS@RPL**

A pointer to the RPL for the I/O request.

For OPEN and CLOSE, points to a dummy full word 0 in protected storage.

For I/O, points to the real RPL. Since the record address and length are provided, and control character prefixing and LOCATE mode (and so on) are handled automatically, there should be little need to actually refer to the RPL.

USER programs may need to refer to the RPL for exotic option flags, KEY pointers, and so on.

The IFGRPL Assembler macro maps this control block.

The parameter list provides all the information that is needed to perform your own I/O processing.

## **Sample USER Program Exit**

The section, *Sample USER Exit—UTIL0038*, on page H-37 in this appendix describes a sample user program, UTIL0038, that is supplied in source form. It illustrates the use of the USER facility to provide a simple encryption/decryption facility.

---

## **Sample FILTER Exit—UTIL0037**

To illustrate the FILTER facility, a sample filter exit program, UTIL0037, is supplied (in source form as well as in compiled form).

This section explains how UTIL0037 is used by filter records.

## UTIL0037 Processing

UTIL0037 filters records by referring to a control table, that is built at OPEN time from a sequential file of control statements.

This file of control statements is named (ddname) by the PARM option of the FILTER operand.

For example:

```
//OUTFILE DD SUBSYS=(NMSS,WTO,  
//          'FILTER=(UTIL0037,FILTCTL)')  
//*  
//FILTCTL DD DSN=MY.FILTER.CONTROL,DISP=SHR
```

Each record to be filtered is actioned against the filter table and accepted or rejected based on strings in the record.

If there are syntax errors in the control file, the attempted OPEN fails and an error message generated that includes the line in error.

## Control File Format

The control file for UTIL0037 is in the following format:

- The control file must be F or FB, LRECL=80. It can be a sequential file or a PDS member (specify the member name on the DD statement for the control file).
- Only columns 1 through 72 of the input record are examined.
- Blank lines are ignored.
- Lines with an asterisk (\*) as the first non-blank character are ignored and thus can be used as comments.
- All other lines must contain valid control statements.

## Filtering Processing

UTIL0037 processes a record to be filtered as follows:

- The record is processed against each statement in the control file in turn.
- Some statements can cause the record to be immediately accepted or rejected. In this case, a return to DD SUBSYS is made with the appropriate return code (0 for ACCEPT, 4 for REJECT).
- If the record reaches the bottom of the control file, then it is implicitly accepted.

## Control Statements

In the following descriptions, *string* is a character string containing any characters (including unprintable ones), delimited by a pair of one of the following characters: /, \, ¢, and |.

Whatever opening delimiter is used, it cannot appear in the string and must be the closing delimiter. The closing delimiter must be followed by a blank.

The following control statements are recognized:

### **BASE *n***

Provides a way to set a logical column number.

By default column 1, as used by the other statements, corresponds to the first byte of a record, unless the file is open for output and has control characters. In this case, column 2 is used as logical column 1.

By specifying **BASE *n***, you nominate which column (*n*) of the record is to be treated as column 1.

#### **Note**

Since the other statements do not allow a column number less than 1, you cannot back up to columns before the logical column set by the **BASE** statement.

The **BASE** statement affects any following statements, and you can have several **BASE** statements in the control file.

There is a default **BASE 1** or **BASE 2** statement assumed at the start of the control file (the position depends on the presence of a control character).

**REJECT** *string* [ *scol* [ *ecol* ] ]

**ACCEPT** *string* [ *scol* [ *ecol* ] ]

**SELECT** *string* [ *scol* [ *ecol* ] ]

Describes a set of records that are to be rejected, accepted, or selected by filtering.

For REJECT, if the match criteria is satisfied, then the current record is immediately rejected by filtering.

For ACCEPT, if the match criteria is satisfied, then the current record is immediately accepted by filtering.

For SELECT, if the match criteria is not satisfied, then the current record is immediately selected by filtering.

If neither *scol* or *ecol* are provided, then the string must match in the current logical column 1 (see *BASE n*, on page H-35).

If just *scol* is provided, then it is either the starting logical column for the string to be in (a number from 1 to 32760), or it can be an asterisk, meaning anywhere in the input record (from logical column 1 onward).

If *ecol* is provided, it sets a logical column range for the string to occur in. *ecol* cannot be specified if *scol* is an asterisk. *ecol* can be either a number (greater than or equal to *scol* plus the length of the string minus 1) or an asterisk, meaning anywhere from *scol* to the end of the record. Note that *ecol* sets the ending column position for the end of the string. So ACCEPT /XYZZY/ 10 20 says search for 'XYZZY' starting in columns 10, 11, 12, 13, 14, and 15.

**REJECT \***

**ACCEPT \***

**SELECT \***

Provides a way of altering the default action at the bottom of the table. By default, a record that reaches the bottom of the table is accepted because it has passed the filtering process. However, you might want to reject all records that pass the filtering process.

Specifying ACCEPT, REJECT, or SELECT with an asterisk instead of a string means: match everything and perform the action.

By default, an ACCEPT \* is generated. However, a REJECT \* at the end of the table discards (rejects) all records that get that far. Do not code any other statements after a REJECT \*, SELECT \*, or ACCEPT \*, because they will never be actioned.

## Sample Filter Table

Following is a sample filter table that illustrates the use of the statements:

```
*
* SAMPLE FILTER TABLE FOR UTIL0037
*
* FILTER OUTPUT FROM IEBCOPY TO SHOW ERROR MSGS ONLY.
*
*
* NOTE THAT THIS IS FOR A PRINT FILE. WE USE THE DEFAULT BASE WHICH
* MEANS THAT COLUMN 2 (AFTER CTL CHATS) IS LOGICAL COLUMN 1.
*
*
* REJECT PAGE HEADERS AND STATEMENT ECHOES (BLANK COL 1)
*
REJECT / PAGE / 110
REJECT / /      1
*
* SELECT END-OF-JOB MSG
*
ACCEPT /IEB147I/
* SELECT ANY E MSGS (3 OR 4 DIGIT NUMBERS)
ACCEPT /E/ 7
ACCEPT /E/ 8
* REJECT THE REST
REJECT *
```

---

## Sample USER Exit—UTIL0038

To illustrate the USERx facility, a sample user exit program, UTIL0038, is supplied (in source form as well as compiled).

This section discusses the use of UTIL0038 (as supplied) to encrypt and decrypt data.

### UTIL0038 Processing

UTIL0038 processes in two modes, depending on whether it is being used for input (USERI), or output (USERO). If called by USER (not USERI or USERO), the attempted OPEN fails.

## UTIL0038 USERI Processing

When being used to process input, UTIL0038 reads from a dataset (the ddname supplied in the PARM operand), decrypts records, and provides them to the application as input.

The ddname named on the PARM operand must reference a dataset that was written as an output file by some other invocation of UTIL0038.

The DCB attributes for the dataset that UTIL0038 reads are RECFM=VBS, LRECL=32760, BLKSIZE=6233. These are the same attributes that it writes.

The DCB attributes for the DD SUBSYS dataset are irrelevant. However, if records of an incorrect length are returned, I/O errors result.

## UTIL0038 USERO Processing

When used to process output, UTIL0038 writes to a dataset (the ddname supplied as the value for the PARM operand), encrypting the written records (that is, those provided by the application as output).

The ddname supplied as the value for the PARM operand *must* reference a dataset. The DCB attributes of this dataset are forced to RECFM=VBS, LRECL=32760, BLKSIZE=6233.

The DCB attributes for the DD SUBSYS dataset are irrelevant. UTIL0038 writes variable length records to its output file.

## UTIL0038 Encryption

The encryption logic used is very simple, and is only supplied to illustrate what is possible.

Encryption is to simply XOR (exclusive OR) the record against a table that contains the values X'FF' down to X'00'. This operation is reversible by simply redoing the XOR.

### Note

This encryption technique is not foolproof. *Do not* attempt to use this program to actually provide data security.

## Notes

Examining the source of UTIL0038 will show how to easily write USERI or USERO programs. It illustrates how to write the exit to be reentrant, how to use the various parameters supplied, how to return OPEN errors, and so on.



---

# Sysplex Support

This appendix describes the following features:

- Cloning support
- Registration with the sysplex automatic restart manager
- Restart status messages

**Note**

Only OS/390 systems support these features.

---

## Cloning Support

The OS/390 system enables you to clone systems by sharing dataset members. This product uses this feature to enable you to clone regions on different systems in a sysplex environment.

To clone regions on different systems, update the RUNSYSIN member for sharing by the regions as follows:

- Step 1. Update appropriate parameter values and dataset names to use defined system symbols to enable the member to be shared between regions on different systems. Useful system symbols include &SYSCLONE, &SYSNAME, and &SYSPLEX.  
You can use the VARxxx control statement to set up your own variables.
- Step 2. Add **SUBS=YES** before the statements in which system symbols or defined variables are used, to enable substitution.

---

## Registration of a Region With the Sysplex Automatic Restart Manager

The sysplex automatic restart manager restarts a registered region automatically if that region fails.

Use the following parameters in a PPREF statement in the RUNSYSIN member to register the region with the restart manager:

**XOPT**=(...,{**NOARM**|**ARM**},...)

Specifies whether the region should (**ARM**) or should not (**NOARM**) register with the restart manager.

**ARMNAME**=*element-name*

Specifies the name used to register the region with the restart manager. The default is *SVM\_acb-name*, where *acb-name* is the value of the **PRI** parameter.

The type of the registered element is **SOLVEMS**.

---

## Restart Status Messages

Restart status messages provide information about the restart status of resources that are registered with the automatic restart manager.

An SSI region can use the event notification facility (ENF) to listen for automatic restart manager events and generate WTO messages for these events. All regions on the same system receive these generated messages.

You can use these messages, and system messages IXC392I and IXC807I through IXC813I, to track the status of the regions that are under the control of the automatic restart manager.

### Enabling the Generation of the Restart Status Messages

To generate the restart status messages, an SSI region must be authorized to listen to automatic restart manager ENF events. You should set up one and only one SSI region per system to listen to these events.

To set up an SSI region as an ENF listener, add the following parameters in the *dsnpref.SIvvv.SSIPARM(SSIPARMS)* member:

**ENF=YES**  
**ENFARMWTO=YES**

## Restart Status Message Syntax

A generated restart status message is of the following form:

NS4U $nn$       ARM *element-status* R: *restart-flag* J: *resource-name*  
EL: *element-type* OS: *old-system* NS: *new-system*  
RG: *restart-group-name*

### NS4U $nn$

Is the ID of the message.  $nn$  can have one of the following values:

<i>nn</i>	<i>element-status</i>	Description
01	RG	The job or started task has registered as an element of the automatic restart manager.
02	RD	The element is ready.
03	DR	The job or started task has deregistered from the automatic restart manager.

### *element-status*

Is the status of the element by which the job or started task is registered with the automatic restart manager. The values are described in the above table.

### *restart-flag*

Indicates whether the job or started task has been restarted. The value can be either N (no) or Y (yes).

### *resource-name*

Is the name of the job or started task.

### *element-name*

Is the name of the element by which the job or started task is registered with the automatic restart manager.

### *old-system*

Identifies the previous system on which the job or started task was active.

### *new-system*

Identifies the current system on which the job or started task is active.

### *restart-group-name*

Identifies the restart group, if applicable, to which the job or started task belongs.

---

# Glossary

**ACB (access method control block)**

The ACB is a control block that links an application program to an access method such as IBM's VTAM.

**Activity Log**

The activity log is a log of all important activities occurring in a region. You can use the log to assist you in determining the cause of a problem.

**Actual State**

The actual state is the state a defined resource is actually in at any given time. In the AUTOMATED mode, ResourceView keeps the actual state consistent with the desired state.

The valid actual states are: ACTIVE, DEGRADED, FAILED, INACTIVE, RECOVERED, STARTING, STOPPING, and UNKNOWN.

**AIM (advanced information manager)**

The advanced information manager is an integrated online database system that interfaces with the MSP operating system.

**Alert**

An alert is an event displayed on the alert monitor. An alert can be internally generated (for example, when a monitored resource fails) or user-defined.

**AOM (Advanced Operation Management)**

The AOM is a facility that manages and controls local and remote operating systems. This is the subsystem interface to a region supported by Automation Services. The interface enables the passage of system messages, events, and commands.

**APF (authorized program facility)**

The authorized program facility identifies programs that are authorized to use restricted system functions.

**API (application program interface)**

The application program interface is a functional interface supplied by a program. The interface enables another application, written in a high-level language, to use specific data or functions of the program.

**APPC (advanced program-to-program communications)**

APPC is an IBM-defined application-level protocol (which makes use of SNA's LU 6.2) that allows interconnected systems to communicate and share the processing of programs.

**AUTOMATED Mode**

The AUTOMATED mode is a service and resource operation mode that allows you to automate many of the functions previously done by operators at consoles. Other modes of operation are OFF, MANUAL, and IGNORED.

**Availability Map**

When a region is implemented, each resource or service defined in the knowledge base can be attached to an availability map. This map indicates the time that the resource or service is available to the system.

**BDAM (basic direct access method)**

The basic direct access method is used to retrieve or update particular blocks of a dataset on a direct access device.

**Boolean**

Boolean is an adjective that describes a logical entity, operator, operation, or expression, and is named after the philosopher George Boole.

**Boolean Expression**

A Boolean expression is a logical expression that evaluates to either false or true.

**CDRM (cross-domain resource manager)**

The cross-domain resource manager provides the SSCP with descriptions and addresses for all the SNA resources available in other domains.

**Child**

A child is an entity that has one or more parents. A child *cannot* start in an automated startup sequence until all the parents have become active.

**CICS (Customer Information Control System)**

The Customer Information Control System is an IBM-licensed program that enables transactions entered at remote terminals to be processed concurrently by user-written application programs. It also includes facilities for building, using, and maintaining databases.

**CNM (communication network management) interface**

The communication network management interface provides an access method to application programs for handling data and commands associated with communication system management.

**Consolidated Console**

The consolidated console enables you to view messages from linked operation regions at a single monitor.

The link between regions occurs at the event level, where INMC enables one region to ship messages to another region.

**CPU (central processing unit)**

The central processing unit is the part of a computer that includes the circuits that control the interpretation and execution of instructions.

**Cross-domain Resource**

A cross-domain resource is a network resource that is owned by the VTAM in another domain.

**CSD (CICS system definition)**

The CICS system definition is a dataset that contains a resource definition record for every record defined to CICS.

**DASD (direct access storage device)**

A direct access storage device is a storage device in which the access time is effectively independent of the location of the data (for example, a disk drive).

**DBCTL (Database Control)**

Database Control is an interface between CICS/ESA and IMS/ESA that allows access to IMS DL/I databases from one or more CICS regions without the need for data sharing. It provides release independence, virtual storage constraint relief, operational flexibility, and failure isolation.

**DCT (destination control table)**

The destination control table contains entries for the transient data queues used by a CICS region.

**Desired State**

The desired state is the required state of a resource. Desired state can be overridden by an availability map or manually from a monitor.

The valid values are ACTIVE and INACTIVE.

**Display Attribute Tables**

The display attribute tables contain display attribute values for resources and icons on the status and graphical monitors. The logical state of a resource determines its display attributes. Authorized users can customize display attributes.

**DL/I (Data Language I)**

The DL/I is a high-level language interface between an application program and the Information Management System (IMS). The interface provides for data definition, manipulation, and control.

**DOM (delete operator message)**

A delete operator message enables the deletion of a non-roll delete message (NRD) from the message monitor or console.

**Domain**

A domain consists of the set of SNA resources controlled by an SSCP. In terms of implementation, an SSCP is the host access method (VTAM). An SNA network consists of one or more domains.

**EBCDIC (extended binary-coded decimal interchange code)**

The extended binary-coded decimal interchange code provides a coded character set consisting of 8-bit coded characters.

**ENF (event notification facility)**

The event notification facility is an OS/390 component that allows authorized programs to listen for specific system events.

**EPS (EndPoint Services)**

EndPoint Services is a communications access method provided by the subsystem interface (SSI) regions. The method provides communication by using the OS/390 cross-system coupling facility (XCF) in a sysplex environment.

**ESA (Enterprise Systems Architecture)**

Enterprise Systems Architecture defines the conceptual structure and functional behavior of a range of IBM mainframe computers.

**ESF (Expert Systems Foundation)**

ESF is a rule-based facility that automates the handling of messages captured by AOM.

**Exit**

An exit is a user-written program or procedure that can be invoked by a region. The exit provides data to the region and performs actions that the region does *not* directly support but that are required by the installation.

**Graphical Monitor**

The graphical monitor is a facility that enables you to monitor and control resources. The monitor provides an iconic view of groups of resources.

**Heartbeat Feature**

The heartbeat feature enables you to specify a time interval at which the region checks the status of the resource. You use the feature to monitor resources that are liable to change state without an accompanying message.

**IBM TCP/IP**

IBM TCP/IP refers to the IBM TCP/IP stacks SecureWay CS and eNetwork CS.

**Icon**

An icon is a graphical representation of grouped services and resources (for example, all printers in a system image) that is displayed on the graphical monitor. If the icon changes color, an operator can check the resources in the resource group to see what has happened and why.

**Icon Panel**

An icon panel is a predefined panel that contains icons that represent groups of resources. For example, an icon panel might contain an icon for the printers in a system, another icon for the started tasks in a system, and so on.

**ICS (Initialization and Customization Services)**

The Initialization and Customization Services is an AutoAssist facility that helps you set up your region parameters.

**IDCM (Integrated Data Communication Manager)**

The Integrated Data Communication Manager is the base product for communication control in the AIM database system.

**IMS (Information Management System)**

The IMS is a database management system that enhances an operating system to provide database processing in both batch and online environments.

**INMC (Inter-Management Services Connection)**

The INMC is a Management Services component that allows multiple regions to be linked and controlled from a single location.

**Internet**

The internet is a collection of linked networks using TCP/IP and related protocols.

**IRC (interregion communication)**

Interregion communication is the method by which CICS provides communication between a CICS region and another region on the same system. The method is used by MRO.

**IS (information system)**

The information system is a computer system that provides information according to user requests.

**ISC (intersystem communication)**

Intersystem communication is communication between separate systems by means of SNA networking facilities or VTAM application-to-application facilities. It links CICS regions and other regions, and is used for communication between user applications or for executing CICS functions in a remote CICS region.

**ISR (Inter-System Routing)**

The ISR feature enables the routing of information between regions.

**JCL (job control language)**

The job control language is used to identify a job to an operating system and to describe the job requirements.

**JES (job entry subsystem)**

The job entry subsystem is a 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.

**Knowledge Base**

A knowledge base is a database in which you store the policies and procedures that govern the operation of your systems.

**Link Station**

In VTAM, a link station is a named resource within a subarea node that represents another subarea node that is attached by a subarea link. In the resource hierarchy, the link station is subordinate to the subarea link.

**Logical Database**

Every region has its own physical database; but in a multisystem environment, you see databases in the linked regions as one logical database.

The changes made to a definition held in a database are reflected across the databases in all regions, regardless of their physical location.

**Logical Resource**

Some resources function purely to initiate some sort of activity from another resource. For example, you can define a resource purely for the purpose of starting or stopping other resources. Other resources might be logical representations of groups of resources. These types of resources need *not* be physical resources. Define these resources as LOGICAL resource types.

**Logical State**

The logical state is the state of a resource based on its actual state, desired state, and operation mode. The logical state of a resource determines its display attributes, and alerts the operator to changes in resource state. The operator can then take action if necessary.

The valid logical states are: OK, ATTENTION, INERROR, PENDING, STARTING, STOPPING, FAILED, DEGRADED, and UNKNOWN.

**LSR (local shared resource)**

Local shared resource pools are used for buffering I/O to VSAM files. NCL supports this type of processing for user databases (UDBs).

**LU (logical unit)**

SNA introduced the concept of the logical unit (LU). The LU is a type of SNA network addressable unit (NAU) that enables end users to gain access to network resources and communicate with each other.

**LU 0**

LU 0 is an unconstrained SNA protocol that allows the selection of any set of available protocol rules, as long as the two LUs are able to communicate with each other successfully according to the rules chosen. Therefore, all LU types are an implementation of LU 0.

**LU 1**

LU1 is an SNA protocol that is used by line-by-line or typewriter type terminals (for example 3767 and 3770).

**LU 6.2**

LU 6.2 is a protocol that serves as a port into an SNA network. LU 6.2 defines a specific set of services, protocols, and formats for communication between logical processors. LU 6.2 provides presentation services for presentation of data to the end user, transaction services for performing transaction processing on behalf of the end user, and LU services for managing the resources of the LU.

**Macro**

*See Registered Macro.*

**Major Node**

A major node is a collection of network resources that can be activated or deactivated as a group.

**MCS (multiple console support)**

The MCS provides the operator interface in an OS/390 system.

**MDO (Mapped Data Object)**

A Mapped Data Object is any data item that can be represented as a continuous string of bytes in storage.

**Message Profile**

A message profile is a set of criteria that determines which messages are displayed on a message consolidation console.

**MODS (Managed Object Development Services)**

The Managed Object Development Services is a development environment that provides tools for creating and customizing NCL applications.

**MRO (multiregion operation)**

Multiregion operation provides communication between CICS regions in the same processor without the use of SNA network facilities. It enables these regions to communicate with each other, and to share resources such as files, terminals, and temporary storage.

**MSP (Multidimensional System Products)**

MSP is a Fujitsu operating system.

**MVS (Multiple Virtual Storage)**

MVS is an IBM operating system.

**NAU (network addressable unit)**

In SNA, a network addressable unit can be a logical unit, a physical unit, or a system services control point. Each unit has a unique network address that identifies it to other nodes in the network.

**NCL (Network Control Language)**

Network Control Language is a structured interpretive language available in a region. The language enables you to develop logical procedures (programs) for performing specific tasks. NCL contains a wide range of logic, built-in functions, and arithmetic facilities which can be used to provide powerful monitoring and automatic control functions.

**NCL Procedure**

An NCL procedure is a member of the Management Services procedures dataset comprising NCL statements. The NCL statements are executed by an EXEC or a START command specifying the name of the procedure.

**NCL Process**

An NCL process is the NCL task that is invoked by an EXEC or a START command to execute one or more NCL procedures. Each NCL process has a unique NCL process identifier.

**NCP (Network Control Program)**

This program resides within and controls the operation of a communications controller. The NCP communicates with VTAM.

**NDB**

An NDB is a database in VSAM Key Sequenced Dataset (KSDS) format.

**Node Type**

In SNA, node type designates a node according to the protocols it supports and the network addressable units (NAUs) it can contain. Five types are defined: 1, 2.0, 2.1, 4, and 5. Type 1, type 2.0, and type 2.1 nodes are peripheral nodes; type 4 and type 5 nodes are subarea nodes.

**NPF (Network Partitioning Facility)**

Network Partitioning Facility is a Management Services facility that defines the functions a user can perform.

**Non-roll Delete (NRD) Message**

A non-roll delete message is a message on an OCS screen. The message indicates an event of critical importance and does *not* roll off the screen as the screen fills up (for example, a message that requires a reply). You remove an NRD message by deleting the message explicitly or by replying to the message.

**OCS (Operator Console Services)**

The Operator Console Services is a Management Services facility that provides a central point of message monitoring and operational control either locally or for a network of regions.

**Packet**

A packet is a sequence of binary digits, including data and control signals, that is transmitted and switched as a composite whole. The data, control signals and, possibly, error control information are arranged in a specific format.

**Parameter Group**

A parameter group contains parameters that determine the characteristics of a region.

**Parent**

A parent is an entity that has one or more children. A parent *cannot* be stopped in an automated shutdown sequence until all the children have become inactive.

**Peripheral Node**

A peripheral node provides end user access to an SNA network.

**Physical Database**

Every region has its own physical database. The database stores, for example, system image definitions, resource definitions, process definitions, and rule set definitions for that region.

The changes made to a definition held in a database are reflected across the databases in all connected regions, regardless of their physical location.

The fact that each region has its own database means that the region can run in isolation if the links to other regions fail.

**PLT (program list table)**

The program list table is a CICS control table that contains a list of programs. The programs in a PLT can be executed as a group during CICS startup or shutdown, and can be enabled and disabled as a group by a single CEMT transaction.

**PPI (program-to-program interface)**

The program-to-program interface enables programs to exchange data.

**PPO (primary program operator)**

The primary program operator is a facility of VTAM that allows unsolicited network messages to be delivered to an application program for processing.

**Process**

A process is a means of automating a series of commands and actions. For example, the sequence of actions involved in shutting down CICS can be translated into a series of steps in a single process. A macro performs the actions required by each step.

**Prompted Field**

A prompted field is a field that is linked to a list of values. A user can select one of these values to complete the field. Authorized users can edit the lists.

**PSM (Print Services Manager)**

The Print Services Manager enables you to control the physical printing of reports on JES or network printers.

**PU (physical unit)**

Each node (a logical grouping of hardware) in an SNA network is addressed by its PU.

**RACF (Resource Access Control Facility)**

The Resource Access Control Facility is an IBM-licensed program that provides for access control by identifying users to the system and verifying their authority, authorizing access to DASD data sets, logging detected unauthorized attempts to enter the system, and logging detected accesses to protected data sets.

**Registered Command**

A registered command contains instructions to the region about the actions to take.

**Registered Macro**

A registered macro points to an NCL procedure that performs the actions in a process step.

## **Relationships**

There are two types of relationships: parent and child. A relationship determines, for example, the automated startup and shutdown sequences of resources. A parent must be active before a child can be started. A child must be inactive before a parent can be stopped. A parent can have more than one child, and a child can have more than one parent.

## **Resource**

A resource is an entity used to provide a function or, together with other resources, to provide a service (for example, a started task, a printer, or an SNA application).

Resources are defined to a region as part of a system image. A resource definition contains the operations policies and methods for a resource.

## **Resource Group**

A resource group represents a group of resources. You attach resource groups to icons. An operator can issue a command to look at the resources if, for example, an icon changes color. The icon might have an attached resource group that contains, for example, all the printers in a system image.

## **Resource Group Filter**

A resource group filter is a set of instructions that determines which resources are contained in a resource group.

## **Resource States**

A resource defined to a region has three types of states: desired, actual, and logical. The region aims to maintain the resource at the desired state.

The logical state indicates the condition of a resource, based on its desired state, actual state, and operation mode. The logical state determines the display attributes of a resource and alerts the operator to any problems.

## **Resource Template**

A resource template provides predefined values for the definition of a commonly-used resource.

Templates for different resources can be copied to different systems and system images.

## **SDO (service-driven operations)**

The SDO is an operations approach that views the information system from an enterprise-wide perspective and manages the system in terms of the services it delivers to the business.

## **SMF (system management facility)**

The system management facility is a standard feature of MSP and OS/390 that provides the means for gathering and recording information that can be used to evaluate system usage.

**SMS (System Management Service)**

The System Management Service is a VOS3 feature that provides the means for gathering and recording information that can be used to evaluate system usage.

**SNA (Systems Network Architecture)**

The Systems Network Architecture is a set of standards that allows the integration of all the different mainframe hardware and software products into a universal network. The standards describe the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of, networks.

**SNMP (Simple Network Management Protocol)**

SNMP is a protocol for the management of the internet.

**SOLVE**

SOLVE encompasses the services provided by Management Services and Automation Services. For example, the SOLVE PPI is a service provided by the subsystem interface (SSI) in Management Services.

**SSCP (system service control point)**

A system service control point is the SNA element that provides the network management and control functions for a network or a portion of a network.

**Status Monitor**

The status monitor enables you to monitor and control individual resources. The monitor displays resource statuses in line-by-line mode. These statuses are color-coded to alert an operator to changes in resource status. Changes in color are governed by changes to the *logical state* of a resource.

**Status Monitor Filter**

A status monitor filter is a Boolean expression that determines which resources are to be displayed on the status monitor. For example, an operator may use a filter that only displays the printers in a system image.

**Subarea**

A subarea is a set of resources controlled by a subarea node.

**Subarea Node**

A subarea node routes data through an SNA network. Type 4 and type 5 nodes are subarea nodes.

**System Image**

A system image is part of the knowledge base and identifies the resources that are to be managed:

- A local image contains local resources that are managed by the local region. Each managed system has its own image.
- A shared image contains resources that may be managed by different regions. A resource in a shared image may move from system to system.
- A sysplex image contains only sysplex resources such as the automatic restart manager (ARM).

**TCP/IP (Transmission Control Protocol/Internet Protocol)**

The Internet Protocol routes packets across multiple subnetworks (such as LANs and WANs). The Transmission Control Protocol is used on top of IP and provides guaranteed delivery of data across the internet.

**Transient Log**

The transient log is a log of activities associated with a resource that is monitored. One transient log exists for each resource definition loaded in a region and exists as long as the definition remains loaded in the region. The resource definition can, however, direct the log entries to the activity log for storage.

**TSO (Time Sharing Option)**

The Time Sharing Option allows terminal operators to interact directly with computer resources and facilities.

**UAMS (User Access Maintenance Subsystem)**

User Access Maintenance Subsystem is the security component of Management Services that enables the definition of users and the associated user authority.

**UDP (User Datagram Protocol)**

User Datagram Protocol is a packet-level protocol used on top of IP. It enables multiplexing of IP datagrams between different application processes but does not guarantee delivery. It is used by SNMP.

**Variable**

A variable is used to store data that can change. A variable is represented by a word that starts with an ampersand (&), followed by the name of the variable. For example, &A is a variable where A is the name. When &A is processed, it is replaced by the stored value.

**VFS (Virtual File Services)**

The VFS provides a VSAM dataset that is used by an Automation Services region for internal processing activities and as a general database.

**VOS3 (Virtual-storage Operating System 3)**

VOS3 is a Hitachi operating system.

**VSAM (Virtual Storage Access Method)**

The Virtual Storage Access Method is used for processing data files that utilize relative, sequential, and addressed access techniques.

**VTAM (Virtual Telecommunications Access Method)**

The Virtual Telecommunications Access Method is an IBM-licensed program that controls the communication between terminals and application programs in an SNA network.

**WTO (write-to-operator)**

WTO is a user-coded service that enables a message to be written to the system console operator, to inform the operator of errors and unusual system conditions that need correcting.

**WTOR (write-to-operator with reply)**

WTOR is a user-coded service that enables a message to be written to the system console operator, to inform the operator of errors and unusual system conditions that need correcting, and to which a response is expected.

**XCF (cross-system coupling facility)**

The cross-system coupling facility is an OS/390 component that provides functions to support cooperation between authorized programs running in a sysplex environment.

**XRF (extended recovery facility)**

The extended recovery facility is an ESA function that enables you to minimize the effect of failures on the users of a software product.

---

# Index

## Symbols

### \$RECALL API

- about, G-5
- EventView variable
  - retrieving the value of, G-49, G-51
  - setting the value of, G-47

### \$REDBAPI API

- about, G-6
- EventView definitions
  - creating, G-53
  - deleting, G-61
  - listing, G-65
  - retrieving information about, G-63
- field names
  - EventView message rules, G-57
  - EventView rule sets, G-56

### \$RMCALL API

- about, G-3
- Automation Services commands, executing, G-7
- knowledge base, retrieving information from, G-12

- link records, extraneous deleting, G-14
- resource status information, retrieving, G-16

### \$RMDBAPI API

- about, G-3
- field names
  - resources, G-23
  - system images, G-22
- ResourceView definitions
  - creating, G-18
  - deleting, G-32
  - listing, G-38
  - retrieving information about, G-35
- system image definition
  - field values, changing, G-41

### \$RMEVENT API

- about, G-4
- sending a message to a specified resource, G-43

### \$RMPB07S procedure, 2-3

### \$RMSTSET API

- about, G-5
- state of a resource, setting, G-45

## A

- abbreviations, product names, 1-8
- AC (Action) action, applying to a parameter group, 2-11
- access control
  - basic, 1-3
  - resource-based, 1-3
  - See also* controlling access
- ACF2, 3-3
  - SAF, using, 3-4
  - security exit
    - full, sample, 3-4
    - partial, sample, 3-4
- activity logs
  - Management Services, part of, 1-6
  - multiple log files, 6-2
  - reusing, 6-2
  - swapping, 6-2
- administration of Automation Services, 1-5
- administrator, group ID, 3-3
  - security settings, A-2
- alert monitor
  - Administration Menu, 2-4
  - customizing the display for a user, 3-20
  - display format, 8-10
    - accessing, 8-10
    - defining, 8-10
    - specifying, 8-12
    - variables, 8-12
      - name, shortening, 8-12
- Alert Monitor : Custom Trouble Ticket panel, 2-7
- Alert Monitor : Email a Trouble Ticket panel, 2-6
- Alert Monitor : Interface Definition panel, 2-5
- Alert Monitor : Trouble Ticket Data Entry Definition panel, 2-8
- Alert Monitor Profile panel, 3-20
- alerts, definition of, 5-7
- API (application program interface)
  - calling, G-2
  - FILTER exit, for, H-18
  - USER facility, for, H-24
  - what they are, G-2
- APIs, specific
  - \$RECALL, G-5, G-47
  - \$REDBAPI, G-6
  - \$RMCALL, G-3
  - \$RMDBAPI, G-3
  - \$RMEVENT, G-4, G-43
  - \$RMSTSET, G-5, G-45
- application program interface. *See* API
- architecture, peer-to-peer, 1-4
- assisted region parameter setup, 1-3
- authority levels, commands, 3-26
- AutoAssist, 1-3
- Automated Mode Attributes Table
  - editing, 7-4
  - example, 7-4
- automatic logon, non-VTAM terminal support, D-3
- automatic problem recording, 2-3
  - \$RMPB07S procedure, 2-3
  - link definitions in SOLVE:Problem, 2-4
  - user ID requirements in SOLVE:Problem, 2-3
- automatic restart manager, I-2
  - restart status messages, I-3
  - syntax, I-4
- Automation Services
  - activity logs, 6-2
  - administration, 1-5
  - commands
    - accessing definitions, 11-3
    - what they are, 11-2
  - component types and class numbers, 3-36
  - databases, 5-6
    - maintenance, 6-3
  - executing commands from an external source, G-7
  - knowledge base, 5-6
  - macro template, 10-2
  - macros, 10-2
  - multisystem operation, 5-6
  - transmitting components, 5-20
  - transmitting service definitions, 5-20
  - what it is, 1-2
- Automation Services API
  - \$RECALL, G-5, G-47
  - \$REDBAPI, G-6
  - \$RMCALL, G-3
  - \$RMDBAPI, G-3
  - \$RMEVENT, G-4, G-43
  - \$RMSTSET, G-5, G-45

## B

- background region, user definition, 3-3
- background users
  - BSYS, 5-15
  - group ID, 3-3
  - security settings, A-7
  - pre-existing definitions, updating, 3-10
- BSYS background user, effect on multisystem implementation, 5-15
- BUSER, background user definition, 3-3

## C

- CHECKALL command and Display Attribute Tables, 7-2
- checkpoint restart, 2-12
  - cold load, 2-12
  - warm load, 2-12
- cloning regions, I-2
- cold load, 2-12
- color, displayed status, 7-3
- column width, status monitor display format, 8-6
- command authority levels, 3-26
- Command Details panel, 11-4
- Command List, 11-3
- command NCL procedures, available variables, 11-7
- Command Prompting, Confirmation and Validation panel, 11-5
- commands
  - adding definitions, 11-6
  - authority levels, 3-26
  - defining your own, 11-6
  - deleting definitions, 11-6
  - executing via the \$RMCALL API, G-7
  - maintaining definitions, 11-6

- registered, 5-7
- registering, 11-3
- what they are, 11-2
- commands, specific
  - CHECKALL, 7-2, G-10
  - GLOBAL, G-11
  - ILOG, 2-2
  - LOAD, G-11
- Confirm Database Synchronization panel, 5-12
- Confirm Unlink panel, 5-19
- connected regions, keeping track of, 5-16
- considerations
  - multisystem implementation, 5-6
  - problem ticket data entry definition, 2-9
- Console Consolidation Profile panel, 3-21
- consoles
  - enabling MASTER authority, C-2
  - extended MCS, C-2
    - migration IDs, C-3
  - JES, C-2
  - management, C-4
    - acquisition, C-4
    - release, C-4
  - number of, C-5
  - OP1, C-2
  - OP2, C-2
  - pool, C-4
  - pseudo, C-2
  - types used
    - Automation Services, by, C-1
    - MSP environment, in, C-2
    - OS/390 environment, in, C-2
    - VOS3 environment, in, C-2
- consolidated console profile, customizing for a user, 3-21
- controlling access
  - functions, using an external security package, 3-34
  - logon through UAMS, 3-3
  - resources, using an external security package, 3-34

- controlling access by using NPF to
  - commands, 3-29
    - Automation Services, 3-30
    - Management Services, 3-30
    - system, 3-30
  - commands issued against resources, 3-31
  - commands issued against systems, 3-31
  - definitions by classes, 3-28
  - definitions by resources, 3-29
  - definitions by system images, 3-28
  - functions, 3-5, 3-26
  - initialization and customization
    - parameter groups, 3-33, 3-42
  - knowledge base, 3-27
  - menu options, 3-27
  - resources, 3-5, 3-26
  - system images, 3-29
- controlling access by using SAF to
  - commands
    - Automation Services, 3-39
    - Management Services, 3-41
    - system, 3-41
  - knowledge base, 3-35
  - menu options, 3-35
  - system images, 3-39
- CPU consumption, C-5
- customizing
  - access to external applications, 2-13
  - display attribute tables, 7-2
  - logon scripts, 2-14
  - prompt list, 9-2
  - security, 3-11
  - system parameters, 2-2
- customizing user profiles
  - alert monitor display, 3-20
  - consolidated console profile, 3-21
  - graphical monitor display, 3-19
  - message monitor profile, 3-20
  - status monitor display, 3-17
  - VTAM domain consolidation profile, 3-23

## D

- data transmission, system responses, 5-7
- data types
  - alerts, 5-7
  - database maintenance events, 5-7
  - registered commands, 5-7
  - resource status events, 5-7
  - system commands, 5-7
  - unsolicited system messages, 5-7
- database synchronization
  - maintaining, 5-16
  - monitoring, 5-14
- Database Synchronization Status panel, 5-13
- databases
  - backups, 6-5
  - maintenance, 6-3
  - maintenance events, 5-7
  - types
    - icon panel, 5-6
    - knowledge base, 5-6
- DD SUBSYS
  - FILTER Exit API, H-18
  - implementing, H-2
  - overview, 1-6, H-2
  - supported functions, H-6
  - USERx facilities, H-16
  - using, H-4
- dialog, System Initialization in Progress, 2-10
- display attribute tables
  - and the CHECKALL command, 7-2
  - customizing, 7-2
  - types, 7-2
  - what they do, 1-5

## E

- e-mail, problem tickets, 2-5
- EndPoint Services. *See* EPS
- EPS (EndPoint Services), multisystem support in sysplex, 5-5
- event-level multisystem support, 5-3
- EventView
  - SMF record format, E-3
  - storing statistics, E-1

- EventView message rules, using an API to
  - create information about, G-53
  - delete information about, G-61
  - list definitions in the knowledge base, G-65
  - retrieve information about, G-63
- EventView rule sets
  - transmitting, 5-20
  - using an API to
    - create information about, G-53
    - delete information about, G-61
    - list definitions in the knowledge base, G-65
    - retrieve information about, G-63
- EventView variable value
  - retrieving, G-49, G-51
  - setting, G-47
- examples, TERMACCESS definitions, D-4
- exit procedures, NCL, F-2
  - introductory section, F-2
  - main processing section, F-5
  - parameters, F-3
    - keyword format, F-4
    - positional, F-4
  - return codes, F-5
  - structure, F-2
- extended display
  - status monitor display format, effect on, 8-9
- extended MCS consoles, C-2
  - effect on performance, C-5
  - migration IDs, C-3
- external applications access
  - customizing, 2-13
  - logging on, 2-14
- external security packages, 3-2
  - accessing, 3-4
  - ACF2, 3-3, 3-7
  - calling using SAF, 3-7
  - controlling access to functions and resources, 3-34
  - full security exit, 3-4
  - load module, 3-4
  - partial security exit, 3-4
  - RACF, 3-3, 3-7
  - Top Secret, 3-3, 3-7

## F

- features of Automation Services
  - AutoAssist, 1-3
  - multisystem support, 1-4
  - user management, 1-3
- field names
  - \$REDBAPI, used in
    - EventView message rules, G-57
    - EventView rule sets, G-56
  - \$RMDBAPI, used in
    - resources, G-23
    - system images, G-22
- Field Prompt Definition panel, 9-4
- FILTER facility, H-9
  - exit procedure
    - API, H-18
    - sample, H-33
- focal point regions, 1-4
  - knowledge base synchronization, 1-4, 5-8
- functions, controlling access to, 3-5, 3-26

## G

- graphical monitor
  - color of displayed status, 7-3
  - customizing the display for a user, 3-19
    - display attribute table data, using, 1-5
    - color, 7-3
    - highlighting, 7-3
    - intensity, 7-3
- Graphical Monitor Profile panel, 3-19
- group IDs
  - administrator, 3-3
    - security settings, A-2
  - background user, 3-3
    - security settings, A-7
  - monitor, 3-3
    - security settings, A-6
  - network operator, 3-3
    - security settings, A-5
  - operator, 3-3
    - security settings, A-3

## H

- highlighting, displayed status, 7-3

## I

- icon panel database, 5-6
- icons, traffic light, 1-5
- ICS (Initialization and Customization Services), 2-2
- ignore (I) action, 2-11
- implementation considerations,
  - multisystem environment, 5-6
- implementation process, implementing
  - automatic problem recording, 2-3
- implementing
  - automatic problem recording, 2-3
  - DD SUBSYS, H-2
- INI procedure
  - editing, 2-16
  - generating, 2-16
  - propagating customized parameter group records, 2-17
- INIFILE parameter, 2-17
- initial dialog, 1-3
- initial program load process. *See* IPL process
- initialization
  - failed parameter groups, listing, 2-11
  - failures, dealing with, 2-10
  - System Initialization in Progress dialog, 2-11
- Initialization and Customization Services.
  - See* ICS
- intensity, displayed status, 7-3
- IPL process
  - Automation Services, accessing during, D-1

## J

- JES consoles, C-2

## K

- knowledge bases
  - installing, 5-6
  - linked, 5-9
  - linking, 5-10
  - multisystem, 1-4
  - staging files, 5-16
  - synchronization
    - focal point regions, 1-4, 5-8
    - subordinates, 1-4, 5-8

## L

- links, multisystem support, 5-4
- List Definition List panel, 8-3
- List Description panel
  - alert monitor display format, 8-11
  - status monitor display format, 8-5
- List Entry Line Fields panel, 8-8
- List Format panel, 8-6
- lists
  - command definitions, 11-3
  - parameter group, 2-11
- logging on to external applications, 2-14
- Logical State Attributes Table, editing, 7-3
- logical state mapping
  - AUTOMATED operation mode, 7-4
  - MANUAL operation mode, 7-5
- logon script, customizing, 2-14
- logs
  - activity, 1-6, 6-2
  - multiple log files, 6-2
  - initialization and customization, 2-2

## M

- Macro Definition panel, 10-4
- Macro List panel, 10-3
- macros
  - accessing definitions, 10-3
  - adding definitions, 10-4
  - deleting definitions, 10-4
  - maintaining definitions, 10-4
  - registering and maintaining, 10-3
- Management Services, 1-6
  - automatic restart manager support, I-2

- Manual Mode Attributes Table
  - editing, 7-5
  - example, 7-5
- MCS (multiple console support),
  - extended consoles, C-2
- message monitor
  - customizing a user profile, 3-20
  - issuing MS commands under NPF, 3-32
- Message Monitor Command Control
  - Profile panel, Initial Command field, 3-20
- messages
  - automatic restart manager, I-3
  - restart status, I-3
  - syntax, I-4
- messages and codes database, 1-6
- migration IDs, extended MCS consoles, C-3
- monitor user, group ID, 3-3
  - security settings, A-6
- monitors
  - graphical monitor, 1-5
  - status monitor, 1-5
- MSP environment, suitable console types, C-1
- multiple console support. *See* MCS
- multisystem environment, knowledge bases, 1-4
- multisystem support
  - BSYS background user, 5-15
  - considerations, 5-6
  - event-level, 5-3
  - focal point regions, 1-4
  - how it works, 5-2
  - links, 5-4
  - overview, 1-4
  - resource-level, 5-3
  - service-level, 5-3
  - subordinates, 1-4
  - sysplex, 5-5
  - types of data, 5-7
  - unlinking regions, 5-19
- Multi-System Support Menu, 5-10

## N

- NCL (Network Control Language)
  - coding your own command procedures, 11-7
  - management services, part of, 1-6
  - variables available to command procedures, 11-7
  - writing procedures to be used as macros, 10-2
- NCL exit procedures, F-2
  - introductory section, F-2
  - main processing section, F-5
  - parameters, F-3
    - keyword format, F-4
    - positional, F-4
  - return codes, F-5
  - structure, F-2
- Network Control Language. *See* NCL
- network operator, group ID, 3-3
  - security settings, A-5
- network partitioning facility. *See* NPF
- NMSSI, H-4
  - common functions, H-7
  - COPY facility, H-7
  - DD SUBSYS support, H-4
  - FILTER facility, H-9
  - PPIRECV facility, H-14
  - PPISEND facility, H-12
  - sample screen, D-7
  - TERMACCESS parameters, D-4
    - examples, D-4
- NMSSI Menu, D-8
- NMSSI program, D-2
  - TERMINALS facility, D-3
- non-VTAM terminal support
  - automatic logon to region, D-3
  - local terminal, D-2
  - regions
    - access definitions, D-3
    - accessing, D-8
  - sessions
    - status, D-9
    - terminating, D-9
  - sysplex, D-2
  - SYSREQ key, D-8
  - Telnet, D-11

NPF (network partitioning facility)

- controlling access to
  - Automation Services commands, 3-30
- commands, 3-29
- commands issued against resources, 3-31
- commands issued against systems, 3-31
- definitions by classes, 3-28
- definitions by resources, 3-29
- definitions by system images, 3-28
- functions, 3-5, 3-26
- initialization and customization parameter groups, 3-33, 3-42
- knowledge base, 3-27
- menu options, 3-27
- MS commands, 3-30
- resources, 3-5, 3-26
- system commands, 3-30
- system images, 3-29
- controlling the types of access to databases, 3-28
- relationship with UAMS, 3-5
- resource list members, 3-5, 3-26
- resource tables, 3-5, 3-26
  - changing, 3-33

## O

OP1 consoles, C-2

OP2 consoles, C-2

operator, group ID, 3-3
 

- security settings, A-3

OS/390 environment, suitable console types, C-1

## P

panels, alert monitor
 

- Administration Menu, 2-4
- Alert Monitor : Custom Trouble Ticket, 2-7
- Alert Monitor : Email a Trouble Ticket, 2-6
- Alert Monitor : Interface Definition, 2-5
- Alert Monitor : Trouble Ticket Data Entry Definition, 2-8

panels, specific
 

- Alert Monitor Profile, 3-20
- Command Details, 11-4
- Command Prompting, Confirmation and Validation, 11-5
- Confirm Database Synchronization, 5-12
- Confirm Unlink, 5-19
- Console Consolidation Profile, 3-21
- Database Synchronization Status, 5-13
- Field Prompt Definition, 9-4
- Graphical Monitor Profile, 3-19
- List Definition List, 8-3
- List Description
  - alert monitor display format, 8-11
  - status monitor display format, 8-5
- List Entry Line Fields, 8-8
- List Format, 8-6
- Macro Definition, 10-4
- Macro List, 10-3
- Multi-System Support Menu, 5-10
- NMSSI logo, D-7
- Resource Monitor Profile, 3-17
- VTAM Domain Consolidation Profile, 3-23

parameter groups. *See* system parameters

PPI (program-to-program interface)
 

- identifying the target receiver, H-12
- source identifier, H-13

PPIRECV, H-14

problem ticket
 

- interface, alert monitor
  - custom, 2-7
  - defining, 2-5
  - e-mail, 2-5
  - implementing, 2-4
- raising, 2-3
- SOLVE:Problem, to
  - \$RMPB07S procedure, 2-3
  - link definitions, 2-4
  - user IDs, 2-3

program-to-program Interface. *See* PPI

prompt lists
 

- adding a value to, 9-2
- disabling variable substitution, 9-3
- greater than signs (>s), 9-4
- less than signs (<s), 9-3
- maintaining definitions, 9-4
- variable substitution, 9-3
- variables in an entry, 9-3

pseudo consoles, C-2

## R

RACF, 3-3

SAF, using, 3-4

security exit

full, sample, 3-4

partial, sample, 3-4

raising a problem ticket, 2-3

regions

initializing and customizing, 2-3

linking, 5-8

BSYS background user, 5-15

non-VTAM terminal access, D-3,  
D-8

automatic logon, D-3

startup confirmation, 4-2

Telnet connections, D-11

unlinking, 5-19

user definition for background  
region, 3-3

Resource Monitor Profile panel, 3-17

resource status events, 5-7

resource-level multisystem support, 5-3

resources

controlling access to, 3-5, 3-26

setting the actual state of, G-45

using an API to

create information about, G-18

delete information about, G-32

list definitions in the knowledge  
base, G-38

retrieve current status

information, G-16

retrieve information about, G-12,  
G-35

send messages to, G-43

ResourceView

SMF record format, E-4

statistics, storing, E-1

## S

SAF (System Authorization Facility)

calling an external security package,  
3-7

controlling access to

Automation Services commands,  
3-39

knowledge base, 3-35

menu options, 3-35

MS commands, 3-41

system commands, 3-41

system images, 3-39

relationship with UAMS, 3-7

sample security templates, 3-8

security

combined NPF and external options,  
3-7

controlling access to MS commands,  
3-41

customizing, 3-11

command replacement, 3-11

updating across linked regions,  
3-11

external options, 3-7

levels, 3-2

UAMS, provided by, 3-3

users, for existing, 3-10

security exit, 3-2

security, controlling access to

Automation Services commands,  
3-30, 3-39

commands, 3-29

commands issued against resources,  
3-31

commands issued against systems,  
3-31

functions, 3-5, 3-26

initialization and customization  
parameter groups, 3-33,  
3-42

knowledge base, 3-27

menu options, 3-27, 3-35

MS commands, 3-30

resources, 3-5, 3-26

system commands, 3-30, 3-41

system images, 3-29, 3-39

the knowledge base, 3-35

- security, supplied group IDs
  - administrator, 3-3
  - background user, 3-3
  - monitor, 3-3
  - network operator, 3-3
  - operator, 3-3
- service definitions, transmitting, 5-20
- service-level multisystem support, 5-3
- ServiceView
  - SMF record format, E-4
  - storing statistics, E-1
- SMF (System Management Facility)
  - EventView record format, E-3
  - header format, E-2
  - record types, E-2
  - ResourceView record format, E-4
  - ServiceView record format, E-4
  - user-defined record format, E-4
- SMFWRITE macro, E-2
- SNA Resource Logical State
  - Normalization Table
    - editing, 7-6
    - example, 7-6
- SOLVE:Problem
  - automatic problem recording, 2-3
  - \$RMPB07S procedure, 2-3
  - link definitions, 2-4
  - user ID requirements, 2-3
- SSI
  - access method for implementing
    - non-VTAM terminal support, D-2
  - startup parameters, D-3
- staging file, 5-15, 5-16
- startup
  - customizing
    - cold load, 2-12
    - warm load, 2-12
  - WTOR confirmation, 4-2
- status
  - color, 7-3
  - non-VTAM terminal support
    - sessions, D-9
- status monitor
  - color of displayed status, 7-3
  - customizing the display for a user, 3-17
  - display format, 8-2
    - accessing, 8-3
    - column width, 8-6
    - defining, 8-3
    - extended display, allowing for, 8-9
    - headings, 8-7
    - multiscreen, 8-9
    - specifying, 8-6
    - variables, 8-7
      - name, shortening, 8-8
  - extended display, 3-18
  - using display attribute table data, 1-5
    - color, 7-3
    - highlighting, 7-3
    - intensity, 7-3
  - views, 8-3
- structured strings, 3-5
- subordinates, 1-4
  - knowledge base synchronization, 1-4, 5-8
- synchronizing databases
  - linking regions, and, 5-8
  - maintaining synchronization, 5-16
- sysplex
  - automatic restart manager, I-2
    - restart status messages, I-3
    - syntax, I-4
  - non-VTAM terminal support, D-2
- SYSREQ key, D-8
- System Authorization Facility. *See* SAF
- system command data flow, 5-7
- system images
  - active, 5-24
  - definition, 1-2
  - developing in isolation, 5-24
  - transmitting, 5-20
  - using an API to change information
    - about, G-41
- System Management Facility. *See* SMF

- system parameters
  - applying actions to parameter groups, 2-2
  - customizing, 2-2
  - listing parameter groups that failed initialization, 2-11
  - parameter group lists, 2-11
  - parameter groups, 2-2, B-2
  - types, 2-2
- system programming, facilities provided by Automation Services, 1-5
- system response data flow, 5-7

## T

- Telnet
  - connections, D-11
  - non-VTAM support, D-11
- TERMACCESS parameters, D-4
  - examples, D-4
- terminals, non-VTAM
  - attaching to and detaching from NMSSI, D-5
  - controlling though customization parameters, D-6
  - defining names, D-5
  - initializing, D-3
  - local, D-2
  - Telnet, D-11
- Top Secret
  - SAF, using, 3-4
- transmitting
  - Automation Services components, 5-20
  - Automation Services service definitions, 5-20
  - EventView rule sets, 5-20
  - knowledge base records, 5-21
- trouble tickets. *See* problem ticket

## U

- UAMS (User Access Maintenance Facility)
  - calling
    - NPF, 3-5
    - SAF, 3-7
  - controlling signon access, 3-3
  - copying user IDs, 3-25
  - relationship with
    - NPF, 3-5
    - SAF, 3-7
  - synchronizing updates across linked regions, 3-11
    - focal point regions, 3-11
    - subordinate regions, 3-11
    - troubleshooting, 3-13
  - update report, 3-15
- unlinking regions, 5-19
- unsolicited system message traffic, 5-7
- updating
  - knowledge bases, 5-16
  - pre-existing background user definitions, 3-10
- User Access Maintenance Subsystem. *See* UAMS
- USER facility
  - API, H-24
  - exit procedure, sample, H-37
- user IDs
  - copying, 3-25
  - defining, 3-25
    - new user, 3-25
  - generating background users, 3-9
  - synchronization, troubleshooting, 3-13
  - user description panels, 3-25
- user profiles
  - automatic user ID generation, 3-25
  - copying, 3-15
  - defining, 3-15
  - deleting, 3-24
  - maintaining, 3-23
  - updating, 3-24
  - user description panel, 3-16
- user-defined SMF record format, E-4

## V

- variables
  - &ZRMCMDDPARMS, 11-7
  - alert monitor display format, for, 8-12
  - command NCL procedures, available to, 11-7
  - EventView
    - retrieving the value of, G-49, G-51
    - setting the value of, G-47
  - global, 2-2
  - status monitor display format, 8-7
    - name aliases, 8-8
- views
  - names, 8-4
  - status monitor, 8-3
- VM programmable operator interface, 3-26

VOS3 environment, suitable console types, C-1

VTAM Domain Consolidation Profile panel, 3-23

VTAM domain consolidation profile, customizing for a user, 3-23

VTAM, communicating with terminals when unavailable, D-1

## W

warm load, 2-12

write-to-operator. *See* WTO

WTO (write-to-operator)

DD SUBSYS operands, H-11

invoking the facility, H-6

using the facility, H-10