CA OPS/MVS® Event Management and Automation

User Guide

Release 12.1

CA Technologies Product References

This document references the following CA Technologies products:

- CA 1® Tape Management (CA 1)
- CA 7™ Workload Automation (CA 7)
- CA 11™ Workload Automation Restart and Tracking (CA 11)
- CA ACF2™ for z/OS (CA ACF2)
- CA Automation Point
- CA Datacom® Database (CA Datacom)
- CA DISK
- CA DRAS
- CA Jobtrac® Job Management (CA Jobtrac)
- CA MIC Message Sharing (CA MIC)
- CA MIM™ Resource Sharing (CA MIM)
- CA NetMaster™ Network Automation (CA NetMaster)
- CA OPS/MVS® Event Management and Automation (CA OPS/MVS)
- CA PDSMAN® PDS Library Management (CA PDSMAN)
- CA Scheduler® Job Management (CA Scheduler)
- CA Spool™ (CA Spool)
- CA SYSVIEW® Performance Management (CA SYSVIEW)
- CA TLMS® Tape Management (CA TLMS)
- CA Top Secret® for z/OS (CA Top Secret)
- CA XCOM™ Data Transport® (CA XCOM)

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Documentation Changes

The following documentation updates have been made since the last release of this documentation:

Note: In PDF format, page references identify the first page of the topic in which a change was made. The actual change may appear on a later page.

- Updated the Substituting Data (see page 222) section.
- Added the Verification and Diagnostic Commands (see page 295) section.
- Updated the Required REXX Programs (see page 284) section.
- Updated the Fields on the SSM Snapshot Panel (see page 239) section.
- Updated the Step 4: Set the Parameters (see page 279) section
- Updated the Moving Resources (see page 275) section.
- Updated the Step 5: Auto-Enable Rules (see page 281) section.
- Updated the Step 6: Add SSM Resource Table Columns (see page 286) section.
- Updated Step 5: Auto-Enable Rules (see page 281) section.
- Updated the CA OPS/MVS Components (see page 649) section.
- Added CMDALL (see page 697) component.
- Added CMDALL (see page 662) rule.
- Added CMDALLC (see page 674) component.
- Added CMDALLR (see page 674) component.
- Added CMDVARY (see page 674) component.
- Added DSNP002I (see page 677) rule.
- Added DSNT376I (see page 677) rule.
- Added DSNT501I (see page 677) rule.
- Added DSNT378I (see page 660) component.
- Added DSNT376I and DSNT501I (see page 660) components.
- Added EMAILMSG (see page 699) component.
- Added EMAILMSG (see page 663) rule.
- Updated EMAILTXT (see page 699) component.
- Updated EMAILTXT (see page 663) rule.
- Added IEA611I (see page 678) component.
- Added IEA611I and IEA794 (see page 665) components.

- Added IEA794I (see page 678) component.
- Updated IPLINFO (see page 663) component.
- Added OPS44020 (see page 665) component.
- Added RESETOSF (see page 708) component.
- Added SSM2XCEL (see page 712) component.
- Added SSM2XCEL (see page 653) rule.
- Added SYSVALRT (see page 713) component.
- Added SYSVCHCK (see page 714) component.
- Added SYSVCTDQ (see page 714) component.
- Added SYSVCTDQ (see page 656) rule.

Contents

Chapter 8: Using SSM Global Application 271

Chapter 9: Using Group Manager 297

Chapter 13: External Product Interface 463

Appendix A: Supplied Sample Rules and Programs 647

Chapter 1: Introduction

This section contains the following topics:

How to Use This Guide

This guide provides information and data that can help you learn how to automate your data center with the CA OPS/MVS product. It provides this information with varying levels of complexity and detail.

The target audience for this guide is any data center automation personnel. It is written so that data center automaters with varying degrees of experience and knowledge can use its content.

Portions of this guide are written sequentially so that users with little knowledge or experience can start at the beginning with CA OPS/MVS basics and, in order of lessons, learn about topics with ever increasing complexity. This also enables users with higher, varying levels of system automation proficiencies to learn about individual, selected topics as single, usable portions of the whole.

Other portions of this guide conform to the more standard, user-guide format. They present reference information about CA OPS/MVS facilities that enable you to powerfully and efficiently design and implement your own system automation blueprint.

If you have little or no experience, start at the beginning of the chapter “How to Begin Using the Product” and proceed as far as you are comfortable. If you already have experience with CA OPS/MVS and want to learn about a specific tool or concept that is explained with a step-by-step lesson, choose that topic from the table of contents (or index) and follow the guide from the point indicated, as long as appropriate. Note that, if appropriate, each lesson refers you to the correct reference section with a corresponding topic.

Portrait of an Automater

This section discusses the role of the person or persons automating your data center.

Anybody can automate the z/OS system of a data center using CA OPS/MVS. Of course, data center operations, systems programming experience, or both are helpful. In reality, however, it is often not operations experience that makes a successful automater-it is more an understanding of the following:

- How your business operates
- Methods and procedures your business follows on a day-to-day basis
- The stated policies and procedures of your organization
- How your enterprise runs its data center, uses its z/OS operating system, and uses those computer programs that were written to help your business operate

Operations Overview—Historically Speaking

This section explains how CA OPS/MVS has developed with respect to the history of data center operations.

Towards the beginning of computer system history, when programs were run using card decks and each line of program code had its own card, programs (jobs) were run one at time. Output, usually in the form of a printout, was produced at the time a job ran. If a job had errors or problems occurred when it ran, error messages were generally indicated on its printout. In other words, computer operations were sequentially oriented. One job ran at a time, and no complicated interfaces existed. At that time, there were no operations to automate.

Later, computer systems shed the bulky card stack technology and had advanced to the point that more than one job could run at the same time. Also, jobs themselves had advanced. As examples, they could read different types of magnetic media for input and print to more than one printer for output. To accomplish these things, operations personnel needed a means of communicating with the operating system of the computer, the programmers who were running jobs, and other operations-type personnel. This communication was accomplished with the implementation of the system console, the master terminal. Through this console, an operator could do things like start all jobs in a data center or be notified about system errors.

As time went on, computer systems became more powerful and data centers became more complex. Data centers were able to keep track of the information needs of an entire business. To do this, some data centers ran more than one computer or operating system 24 hours a day, seven days a week. And their operators began using multiple system consoles to communicate efficiently with this complex of computer activity.

Today, computers and their operating systems have become so powerful that many thousands of jobs can run every day in one data center. The z/OS operating system for instance, the operating system of choice for many large data centers, can run multiple versions of itself and each can run many jobs at the same time. In these complex data centers, human operators cannot keep up with all system events. No person or persons can.

That is why a system automation product such as CA OPS/MVS has become critical. It can respond automatically to any or all events that occur in a z/OS system such as the following:

- Follow established procedures
- Watch the message stream of the system
- Issue DISPLAY commands and examine their output
- Track time and execute scheduled actions based on time
- Apply common sense
- Take action

- Call for help
- Answer queries about the status of your system

Chapter 2: How to Begin Using the Product

This section contains the following topics:

Overview

This chapter provides a step-by-step approach to learning system automation techniques using CA OPS/MVS. In it, you begin by using the Automated Operations Facility, or AOF, to create a rule that suppresses one type of z/OS system message. Using AOF, we also modify that rule, test it, and enable and disable it. In later lessons, you use other CA OPS/MVS facilities to establish more rules in different ways.

OPSVIEW is the main user interface to CA OPS/MVS. It is through the OPSVIEW network of panels and user menus that you can communicate with and use CA OPS/MVS.

The following OPSVIEW Primary Options Menu is the origination point from which you enter all CA OPS/MVS facilities and functions:

```
CA OPS/MVS ------ CAxx ---  OPSVIEW Primary Options Menu  ----- Subsystem OPSA

 0  Parms      Set OPSVIEW and ISPF default values          User ID  - USER01
 1  OPSLOG     Browse OPSLOG                               Time     - 12:43
 2  Editors    AOF Rules, REXX programs, SQL Tables        Release  - 11.8
 3  Sys Cntl   Display/Modify System Resources             SP       - 0
 4  Control    Control CA OPS/MVS
 5  Support    Support and Bulletin Board information
 6  Command    Enter JES2/MVS/IMS/VM commands directly
 7  Utilities  Run CA OPS/MVS Utilities
 A  AutoMate   CA AutoMate rules edit and control
 I  ISPF       Use ISPF/PDF services
 S  SYSVIEW    CA SYSVIEW
 T  Tutorial   Display information about OPSVIEW
 U  User       User-defined applications
 X  Exit       Exit OPSVIEW

 CA OPS/MVS Event Management and Automation
 Copyright © 2010 CA.  All rights reserved.
```

By following this chapter from beginning to end, you should gain a fundamental knowledge of how an automation rule works.

Ways to Automate

You can use several methods to automate your system and test your automation rules.

The following are the various automation methods and references to corresponding lessons on the following pages:

AOF Edit

AOF Edit (OPSVIEW option 2.1) enables you to create rules using the OPS/REXX programming language.

See Lesson 1: Create a Rule With the AOF in this chapter.

Test and Verify

With CA OPS/MVS, you can test and verify your automation rules before they get into a production environment.

See Lesson 2: Test and Verify a Rule in this chapter.

Establish More Rules

With CA OPS/MVS, you can establish another message suppression rule and establish a test command rule.

See Lesson 3: Establish More Rules in this chapter.

Organizing Rules

CA OPS/MVS gives you the ability to group rules in a meaningful way. These groups are called rule sets.

See Lesson 4: How to Organize Rules into Rule Sets in this chapter.

Enable and Disable Rules and Rule Sets

With CA OPS/MVS, enabling and disabling rules and rule sets is how you turn them on and off.

See Lesson 5: How to Enable and Disable Rules and Rule Sets in this chapter.

EasyRule

EasyRule (OPSVIEW option 2.3) is a fourth-generation facility that makes it easy to create rules that will respond to various system events, including system messages. It provides the ability to modify the display of system console messages without the need for programming.

To create very complex rules, you may need to use the AOF edit facility rather than EasyRule.

See Lesson 6: Solve a Problem Using EasyRule in this chapter.

Automation Analyzer

The Automation Analyzer (OPSVIEW option 7.2) examines and displays a statistical analysis of the message activity of your system. You can use this information to choose the messages you want to suppress, and then create message suppression rules directly from the panels of the Automation Analyzer.

See Lesson 7: Suppress Messages Using the Automation Analyzer in this chapter.

MPF Conversion Facility

Most z/OS installations have used the IBM product for message suppression, called the Message Processing Facility (MPF). CA OPS/MVS provides an MPF conversion facility (OPSVIEW option 7.3) to enable these installations to migrate without losing the time they have invested in MPF. The MPF conversion facility reads messages from the MPF message suppression list and automatically generates CA OPS/MVS rules.

The MPF conversion facility is limited to simple suppression entries. Messages regarding console colors and exit information are not automatically processed. If you use the MPF conversion facility, you must modify parmlib to remove message suppression.

See Lesson 8: How to Create Rules From an MPF Suppression List in this chapter.

Purpose of These Lessons

The first lesson starts your automation education by showing you how to have CA OPS/MVS automatically suppress system messages that you do not want to see. We start with this topic for the following reasons:

- Most system automaters start automating a system by suppressing unwanted messages.
- Rules that govern how message suppression operates can be fairly simple.
- Once a few message suppression rules are established, we can illustrate how several CA OPS/MVS facilities function by investigating how those rules affect your system.

How to Prepare Your System for Your Lessons

For the purposes of the lessons in this chapter, ensure that a normal, fully qualified, partitioned data set without members has been allocated. Such a data set is used in the lessons to act as a repository for automation rules, each member of which is a rule, and for testing rules once they have been established. A good name for this test rule set could be *userid*.TEST.RULES.

For testing, you should always use a test rule set, rather than a production rule set.

- If a Test Rule Set Does Not Exist

  Either proceed to Allocate a Test Data Set (see page 36) to establish your test rules library or ask your systems programmer to do it for you.

- If a Test Rule Set Exists

  Proceed to Purpose of These Lessons (see page 34) in this chapter.

Allocate a Test Data Set

Begin at your ISPF Primary Option Menu. ISPF is the main user interface for your z/OS operating system.

This panel is the origination point from which you enter all ISPF facilities and functions.

To allocate a test data set

1. Enter 3.2 in the Option field.

   The Data Set Utility panel is displayed. Use this panel to enter the name of a data set that you can use for testing rules. Your user ID, with TEST, and RULES, makes a good name for such a data set.

2. Fill in the Data Set Utility panel with the following suggested entries.

   - Enter an A in the Option field.
   - Enter your user ID in the Project field.
   - Enter TEST in the Group field.
   - Enter RULES in the Type field.

   The Data Set Utility panel will look similar to that shown here:

```
                              Data Set Utility
Option ===>
    A Allocate new data set                 C Catalog data set
    R Rename entire data set                U Uncatalog data set
    D Delete entire data set                S Data set information (short)
blank Data set information                  M Enhanced data set allocation
                                            V VSAM Utilities

ISPF Library:
   Project  . . USERID
   Group  . . . TEST
   Type . . . . RULES
Other Partitioned, Sequential or VSAM Data Set:
   Data Set name . . .
   Volume Serial . . .          (If not cataloged, required for option "C")
Data Set Password  . .          (If password protected)
```

3. Review your choices and press enter.

   The Allocate New Data Set panel is displayed, similar to the one shown here:

```
------------------------- Allocate New Data Set --------------------------
Command ===>
                                                                More:     +
Data Set Name . . . : USERID.TEST.RULES
Volume serial . . . . MVSNN0        (Blank for authorized default volume) *
Generic unit. . . . .               (Generic group name or unit address) *
Space units . . . . . TRACK         (BLKS, TRKS, CYLS, KB, MB or BYTES)
Primary quantity. . . 20            (In above units)
Secondary quantity. . 10            (In above units)
Directory blocks. . . 100           (Zero for sequential data set)
Record format . . . . FB
Record length . . . . 80
Block size  . . . . . 3200
Expiration date . . .               (YY/MM/DD, YYYY/MM/DD
                                     YY.DDD, YYYY.DDD in Julian form
 Enter "/" to select option          DDDD for retentions in days
   Allocate Multiple Volumes         or blank)
```

4. Provide the allocation data using the above panel as an example to enter values for the data requested by the operating system. Perform the following actions:

   - Enter data in the fields as shown on the above panel. These values are suggestions only; however, they will work for a rule testing data set.
   - Press your PF3 key twice or enter the END command twice.

Your test data set is allocated and you may proceed to the next section, Purpose of These Lessons.

Some System Messages Create Visual Noise

Every computer application issues messages. They fall into two general categories:

- Messages that call for an operator action, such as replying to a WTOR, calling a customer engineer, or restarting a resource
- Informational messages that do not call for operator action

Messages of the second category can be such an overwhelming part of the console display of the operator that they make it hard for an operator to see and respond to messages of the first category.

You can make the task of the operator more manageable by writing rules to suppress particular messages from displaying on the console, appearing in SYSLOG, or both. If you choose to only *suppress* a message, that message does not completely disappear; it is routed to SYSLOG, where it can be referenced as needed. If you prefer, you can choose to *suppress and delete* a message; it is not only suppressed from the console but also deleted from SYSLOG. However, even if you choose to suppress and delete a message, it is still recorded in OPSLOG, the very powerful and flexible system log provided with CA OPS/MVS.

Lesson 1: Create a Rule Using the AOF

By following the procedures outlined in this first lesson, you will:

- Access the AOF edit facility.
- Use AOF edit to create a suppression rule with the CA OPS/MVS own programming language, called OPS/REXX.

To create a rule using the AOF

1. Access the AOF from the OPSVIEW Primary Options Menu by entering 2.1 in the Option field.

   The AOF Edit Entry panel displays:

```
AOF EDIT - Entry panel --- MSI1 --- OPSVIEW -------------------Subsystem OPSS
COMMAND ===>
RULE LIBRARY:
   PROJECT ===> USERID
   GROUP   ===> TEST         (* for all RULE SETs)
   TYPE    ===> RULES
   MEMBER  ===>               (Blank for MEMBER selection list)
OTHER PARTITIONED DATA SET:
   DATA SET NAME  ===>
-------------------------  AOF TEST DATA  --------------------------------
  (Blank all fields below in order to test with temporary data.)
TEST DATA SET NAME:
   PROJECT ===>
   GROUP   ===>
   TYPE    ===>
   MEMBER  ===>
OTHER PARTITIONED DATA SET:
   DATA SET NAME  ===>
```

2. Specify which rule library to access by typing in the name of your test data set, leaving the Member field blank, and then press Enter.

   The corresponding AOF Test Rule List panel displays:

```
AOF TEST - Rule List ------ USERID.TEST.RULES -------------------------------
COMMAND ===>                                                SCROLL ===> PAGE
   Line Commands:     R EasyRule    S ISPF Edit     T Test    C Compile   V View
   E Enable  D Disable  A Set Auto-Enable Z Reset Auto-Enable   X Delcomp
        Test Start Date   : 2009/10/12  Test Start Time  : 13:54:00
        Test Current Date : 2009/10/12  Test Current Time: 13:54:00
  RULENAME STATUS  AE TYP VV.MM  CREATED  MODIFIED     SIZE INIT  MOD   ID
  **END**
```

   Because this data set is empty, no rules are displayed. If there had been rules (members) in the data set, they would be displayed.

   Besides using this panel to select rules once they exist, you can also create a rule. CA OPS/MVS automatically creates a rule when you select it on the COMMAND line.

3. Create a rule that suppresses system message IEF450I by typing S IEF450I on the command line. When you are finished press Enter.

   A rule named IEF450I is created and the AOF test edit panel appears:

```
AOF TEST ----- USERID.TEST.RULES(IEF450I) - 01.00--------- COLUMNS 000 000
COMMAND ===>                                               SCROLL ===> PAGE
****** *************************** Top of Data ***************************
''''''
''''''
```

   The format of the AOF Test edit panel is nearly identical to the editing display of the ISPF/PDF editor. The two panels also function similarly.

   You use the panel to enter and create a REXX-based message suppression rule.

   For more information about rule names, see the *AOF Rules User Guide*.

4. Enter the text of the REXX program that is your first suppression rule. Your program should look just like the following:

```
AOF TEST ----- USERID.TEST.RULES(IEF450I) - 01.00--------- COLUMNS 000 000
COMMAND ===>                                               SCROLL ===> PAGE
****** *************************** Top of Data ***************************
'''''' )MSG IEF450I
'''''' )PROC
'''''' RETURN 'SUPPRESS'
''''''
''''''
```

   This three-line REXX program is actually a rule that suppresses the system message IEF450I. A rule to suppress a different message would look like the one shown here, except a different message ID would appear in the first line of the rule.

5. Enter END on the command line or press your PF3 key.

   Your rule is created.

Once you finish this part of Lesson 1, your rule is ready to test and verify. You accomplish this by enabling it and applying several different CA OPS/MVS functions to it. You can do all of this from the AOF Test Rule List panel as done in the next lesson Test and Verify a Rule.

Lesson 2: Test and Verify a Rule

By following the procedures in this lesson, you will:

1. Test your new rule.
2. Verify your new rule.

To test and verify a rule

1. Verify that the AOF Test Rule List panel is being displayed:

   If not, proceed with the following instructions:

   - If the AOF Edit panel is being displayed, press your PF3 key (or enter END).
   - If the main OPSVIEW menu panel is being displayed, select 2.1, and then enter your test library name in the AOF EDIT Entry panel.
   - If the AOF Edit Entry panel is being displayed, enter your test library name.

2. Enable your rule by typing E to the left of the name of your rule and then press Enter.

   In response, two fields on the Rule List Panel change, as shown in the following example screen. This example shows how the panel looks after the changes. Notice that for the rule named IEF450I, the value in the Status field has changed to ENABLED and the value in the Typ field has changed to MSG. In addition, the message AOF RULE ENABLED appears in the upper-right corner of the panel.

   Note: For more information about enabling rules, see Lesson 5: How to Enable and Disable Rules and Rule Sets in this chapter.

   Note: If your rule contains syntax errors, the E command fails. If this happens, go back to the AOF panel and ensure that your REXX rule program is exactly like that shown in Lesson 1.

```
AOF TEST - Rule List ------ USERID.TEST.RULES ---------------AOF RULE ENABLED
COMMAND ===>                                                  SCROLL ===> PAGE
   Line Commands:     R EasyRule    S ISPF Edit     T Test    C Compile    V View
   E Enable  D Disable  A Set Auto-Enable Z Reset Auto-Enable   X Delcomp
        Test Start Date   : 2009/10/12  Test Start Time  : 13:54:00
        Test Current Date : 2009/10/12  Test Current Time: 13:54:00
  RULENAME STATUS  AE TYP VV.MM  CREATED  MODIFIED     SIZE INIT  MOD   ID
  IEF450I  ENABLED  Y MSG 01.04 09/10/12 09/10/12 18:38   7    5    3 SYSADM
```

3. Type T to the left of the name of the rule and press Enter.

   When you select a message rule for testing, the AOF takes you to the AOF Test MSG panel, which prompts you for information about the message rule you want to test. A sample panel appears in the next step. At the bottom of this panel, a portion of the OPSLOG is displayed.

4. Enter the following command in the Command field:

   ```
   D TIME DISP
   ```

This command tells CA OPS/MVS to display the following information for each rule:

- Time that the event with which the rule is connected appeared in OPSLOG
- Final disposition of the event as determined by the AOF

```
AOF Test MSG ------------ MSI1 --- OPSVIEW --- 15:27:24  12OCT2009 COLS 001 070
COMMAND ===>                                                  SCROLL ===> PAGE
 REXX Trace ==> N   Live Commands  ==> NO       Access Auto Test Data:   (Y/N)
 Msg Id:             Msg Disp:              Hardcopy Log:
 Jobname     ==>                           IMS Id      ==>
 Job Id      ==>                           Exit Type   ==> MVS
 MSF Sys     ==>                           Console Id  ==>
 User        ==>                           Console Nm  ==>
 Sys Id      ==>                           MCS Flags   ==>
 Special Ch  ==>                           Descriptor  ==>
 Route       ==>
 Term Name   ==>                           Report Id   ==>
 Message     ==>
Time     ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
******** ******************** TOP OF MESSAGES  ******************************
15:27:24 ENABLE TESTING.IEF450I
```

5. Enter values on the AOF Test MSG panel:
   - Type any characters in the Jobname field. Although it does not matter for purposes of this lesson what characters are listed in the Jobname field, a rule test will not run unless that field has been filled. For example, you could type XXXX into the Jobname field.
   - Type the message ID and any text in the Message field. For example, you could type IEF450I SAMPLE MESSAGE into the Message field.
   - Type 0 in the Console Id field.
6. Press Enter.
   - A test message is run against the enabled rule.
   - The following sample panel shows the results of your rule test.
   - The value SUP that appears in the Dis (for disposition) column indicates that message IEF450I was successfully suppressed.

```
AOF Test MSG ------------ MSI1 --- OPSVIEW --- 15:27:24  12OCT2009 COLS 001 070
COMMAND ===>                                                  SCROLL ===> PAGE
 REXX Trace ==> N   Live Commands  ==> NO       Access Auto Test Data:   (Y/N)
 Msg Id:  IEF450I    Msg Disp:  Suppress       Hardcopy Log:
 Jobname     ==> XXXX                      IMS Id      ==>
 Job Id      ==>                           Exit Type   ==> MVS
 MSF Sys     ==>                           Console Id  ==> 0
 User        ==>                           Console Nm  ==>
 Sys Id      ==>                           MCS Flags   ==> 000000
 Special Ch  ==>                           Descriptor  ==> 0000
 Route       ==> 00000000000000000000000000000000
 Term Name   ==>                           Report Id   ==>
 Message     ==> IEF450I SAMPLE MESSAGE
Dis    ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
******** ******************** TOP OF MESSAGES  ******************************
NON ENABLE TESTING.IEF450I
NON ENABLE TESTING.IEF450I
000 OPR39000 RULE TESTING.IEF450I FOR MSG IEF450I          NOW ENABLED
SUP IEF4501 SAMPLE MESSAGE
```

7. Examine the results of the test in the OPSLOG and when satisfied, press PF3 or type END on the command line.

Your rule is tested and verified.

Go to Lesson 3: Establish More Rules.

Lesson 3: Establish More Rules

By following the procedures in this lesson, you can:

- Establish another message suppression rule.
- Establish a test command rule.

To establish more rules

1. Verify that the AOF Test Rule List panel is being displayed:

   If not, proceed with the following instructions:

   - If the AOF Edit panel is being displayed, press your PF3 key (or enter END).
   - If the main OPSVIEW menu panel is being displayed, select 2.1, and then enter your test library name in the AOF EDIT Entry panel.
   - If the AOF Edit Entry panel is being displayed, enter your test library name.

2. Type S IEC233I on the Command line and press Enter.
3. Enter the text of another suppression rule.

   Your program should look just like the one shown here. This four-line REXX program is another rule that suppresses the system message IEC233I.

```
AOF TEST ----- USERID.TEST.RULES(IEC233I) - 01.00--------- COLUMNS 000 000
COMMAND ===>                                               SCROLL ===> PAGE
****** ***************************** Top of Data ****************************
'''''' )MSG IEC233I
'''''' )PROC
'''''' IF MSG.JOBNAME ='J1234' THEN RETURN 'SUPPRESS'
'''''' ELSE RETURN 'DELETE'
''''''
''''''
''''''
''''''
```

4. Press your PF3 key or enter END on the Command line.

   Your newest rule is created and ready to test and verify as you did in Lesson 2: Test and Verify a Rule.

5. Establish a Command Rule by typing S CMDTEST on the Command line of the AOF Test Rule List panel.
6. Enter the text of a new command rule.

   Your program should look just like the following one. This REXX program is a rule that responds to a z/OS command event.

```
AOF TEST ----- USERID.TEST.RULES(CMDTEST) - 01.00--------- COLUMNS 000 000
COMMAND ===>                                               SCROLL ===> PAGE
****** ***************************** Top of Data ****************************
'''''' )CMD CMDTEST
'''''' )PROC
'''''' SAY OPSINFO('CPUID') 'FOR THIS CPU'
'''''' RETURN 'ACCEPT'
''''''
''''''
''''''
''''''
''''''
''''''
```

7. Press your PF3 key or enter END on the Command line.

   Your command rule is created and ready to test and verify as you did in Lesson 2: Test and Verify a Rule.

Go to Lesson 4: How to Organize Rules into Rule Sets.

Lesson 4: How to Organize Rules into Rule Sets

CA OPS/MVS provides a flexible structure for organizing rules into rule sets, so that each data center can use the classification system that best meets its needs.

In a well-designed classification system, rules are grouped in a useful and intuitive way, and rule set names describe the rules they contain. For more information about organizing rules, see the *AOF Rules User Guide*.

The process to organize rules into rule sets is as follows:

1. Choose a rule classification system

   Rules are grouped in a useful and intuitive way. The following two issues are involved in choosing a rule classification system:

   - Reason for grouping rules into a rule set
   - Choice of the rule set name

2. Choose rule set names and contents

   Rule set names describe the rules they contain.

   The following are example rule set names and contents:

   - Rules grouped by production environment into rule sets can be named SYSOP for operators and SYSPROG for system programmers.
   - Rules grouped by application/event into rule sets can be named IPL and WEEKEND.
   - Rules grouped by type of rule into rule sets can be named ACTION, SUPPRESS, and TOD.

3. Choose where to put your rules

   This scenario discusses placing message suppression rules, which are grouped by the type of SUPPRESS.

   - Where to put the first message suppression rules

     If you are installing the first CA OPS/MVS message suppression rules in your system, you need to choose the classification system to use. We suggest you place all message suppression rules into a rule set named SUPPRESS.

   - Where to put more message suppression rules

     If message suppression rules have already been put into your production system, you should follow the rules classification system that has already been established. If an inspection of the rule set list (OPSVIEW option 4.5.1) does not make the classification system clear, then ask the person who first installed message suppression rules for guidance.

Go to Lesson 5: How to Enable and Disable Rules and Rule Sets.

Lesson 5: How to Enable and Disable Rules and Rule Sets

This lesson discusses how to enable or disable rules using the OPSVIEW Primary Options Menu 4.5.1 panels.

Enabled rules respond to system events; disabled rules do not, even if their corresponding events occur.

The following four line commands control whether a rule is enabled:

A

Sets auto-enable. When auto-enable is set for a rule, the rule is enabled every time CA OPS/MVS starts running, and again whenever an Enable command is issued for the rule set in which the rule is stored.

D

Disables the rule.

E

Enables the rule.

Z

Clears auto-enable

Important! Line commands A and Z affect every rule in the rule set, as if you had entered it on every line of the Rule List panel. Since automation strategy often includes selectively enabling rules, changing the status of all the rules in a rule data set may have unintended consequences. There is no easy way to restore the distinction between auto-enabled rules and others once all the rules of a set have been changed.

Line commands A and D sometimes interact with each other to determine whether a given rule should be enabled. Their exact function depends on whether they are issued in the Rule List panel or on the Rule Set List panel.

The following table describes how the commands operate on the Rule List and Rule Set List panels, and the effect they have on rules:

Command	On the Rule List Table	On the Rule Set List Table	Effect on the Rule
Enable	Operates unconditionally to enable rules.	Enables only those members of the rule set that have Auto-enable set.	Has an immediate effect on whether a rule reacts to a system event.

Command	On the Rule List Table	On the Rule Set List Table	Effect on the Rule
Disable	Operates unconditionally to disable rules.	Disables all members of the rule set.	Has an immediate effect on whether a rule reacts to a system event.
Set	Operates unconditionally to set Auto-enable.	Sets Auto-enable for all members of a rule set.	Only controls future enable events and does not affect the current enabled status.
Reset	Operates unconditionally to reset Auto-enable.	Clears Auto-enable for all members of a rule set.	Only controls future enable events and does not affect the current enabled status.

Enable or Disable a Single Rule

If you have not read Lesson 4: How to Organize Rules with Rule Sets, do so now. Then proceed.

To enable or disable a single rule

1. Access the AOF control facility.

   - Access the AOF CTRL panel. Begin at the OPSVIEW Primary Options Menu (shown in Lesson 1: Create a Rule Using the AOF). Type 4.5.1 in the Option field and press Enter. As a result, the AOF CTRL Entry panel appears. A sample panel is shown here:

```
AOF CTRL - Entry panel --- MSI1 --- O P S V I E W ------------ Subsystem OPSS
COMMAND ===>
Rule data sets of the form  OPS.*.RULES:
   Rule sets  ===>              ( * or blank for all rule sets )
       Either specify a specific rules et or request a list of all rule sets.
   Stats    ===> Y            ( Y to list statistics or N to suppress them )
       When listing all rule sets, you can request suppression of cumulative
       statistics and experience faster response by specifying 'N' above.
   System   ===> *LOCAL*                 WAIT ===> 10
       Either specify the name of an MSF connected system or * for the local
       system.  Enter ? for a list of MSF connected systems.
```

   - Access the Rule Set List Panel. On the AOF CTRL Entry panel, type an asterisk (*) in the Rule sets field and press Enter. Or, just press Enter without making an entry.
   - (Optional) Fill in the Stats field before pressing Enter to indicate whether the Rule Set List panel includes the cumulative statistics.

   Note: When requesting the Rule Set List panel, place a Y in the Stats field of the AOF CTRL Entry Panel so that statistics are included.

   In response to your entries, the Rule Set List panel appears, as shown in the following sample:

```
AOF CTRL - Rule Set List ------ MSI1 --- OPS.*.RULES -------------- ROW 1 OF 3
COMMAND ===>                                                  SCROLL ===> PAGE
      Line Commands: S Select   E Enable   D Disable   U Utilities
      A Set Auto-Enable   Z Reset Auto-Enable   C Compile   X Delete Compile
                           System: *LOCAL*
RuleSet  Status  AE CNT  VV.MM Created  Changed        Size Init  Mod   ID
ACTION   ENABLED  Y 2    01.00 09/02/18 09/02/18 13:17  101  101    0 OPSLCD
CICS     ENABLED  Y 2    01.00 09/01/12 09/02/21 15:59   15   20    1 OPSKED
DB2      ENABLED  Y 25   01.00 09/03/30 09/03/30 08:42  237  237    1 OPSRF
```

2. Enable or disable an individual rule. Follow these steps to enable or disable a single rule:

   a. Select the rule set that contains the rule you want. From the Rule Set List panel, select a rule set by typing S to the left of the name of the rule set and pressing Enter.

In response, the following Rule List panel appears listing all of the selected rules in the rule set:

```
AOF CTRL - Rule List ------ MSI1 --- OPS.IMS.RULES            Row 1 to 2 of 2
Command ===>                                                  Scroll ===> PAGE
   Line Commands: R EasyRule   S ISPF Edit   C Compile   X Delete Compile
          V View  E Enable  D Disable  A Set Auto-Enable  Z Reset Auto-Enable
                          System: *LOCAL*
  RuleName Status  AE TYP VV.MM  Created  Changed       Size Init  MOD   ID
  DFS994I  ENABLED  N MSG 01.00 09/11/14 09/11/14 06:32    3    3    0 USER123
  IMSCTRL  DISABLED N *** 01.00 09/11/14 09/11/14 06:33    3    3    0 USER123
```

b. Enable a rule by typing E to the left of the name of the rule and press Enter. To disable a rule on the Rule List panel, type D to the left of the name of the rule and press Enter.

Enable or Disable an Entire Rule Set

This exercise illustrates the interaction between the Set auto-enable and Enable commands.

These steps use Enable and Disable commands on the Rule Set screen to enable or disable all the rules in a rule set. Rule sets do not have a group Enabled indicator. The Enable or Disable operations you use change the settings of all the individual members of the rule data set.

To enable or disable an entire rule set

1. Identify the rule set that you want to enable or disable. Use a test rule set, rather than a production set, to avoid harmful effects on your system automation. On the Rule Set List panel, look for the name of the target rule set. The Status field immediately to the right of each rule set name provides information about the rules in that rule set:
   - ENABLED means that at least one rule in the rule set is currently enabled.
   - DISABLED means that every rule in the rule set is currently disabled.
2. To disable all the rules in the set, type D to the left of the name of the rule set and press Enter.

   Disable works unconditionally on all the rules in the rule data set.
3. To enable all the rules in the set, type A to the left of the name of the rule set and press Enter, then type E to the left of the name of the rule set and press Enter.

   Enable works on rules in the set that have their Auto-enable bit set, therefore, to enable all the rules, ensure that the Auto-enable bit is set for all rules.

Note: This exercise destroys the Auto-enable status of individual rules in the rule set on which you practice. Use a test rule data set for the exercise to avoid damaging a production rule data set.

Lesson 6: Solve a Problem Using EasyRule

Suppose that you want to increase the visibility of NOT CATALOGED 2 conditions. Such conditions occur when a data set cannot be cataloged because a data set with the same name already exists on another volume.

Messages with an ID of IEF287I notify you of NOT CATALOGED 2 conditions, while those with an ID of IEF285I are related normal messages that do not require action. You need to be aware of IEF287I messages, because they are indicative of production problems. However, the presence of IEF285I messages creates visual noise, making it difficult for you to see and respond to the important IEF287I messages.

You can solve the problem by performing these tasks:

- Reword the text of IEF287I messages so that they indicate what the problem is and that CA OPS/MVS is taking actions to correct it.
- Highlight the reworded message text.
- Suppress IEF285I messages.

This lesson describes how you can use EasyRule to create two rules to perform these tasks.

- The first rule, called NOTCTLG (see page 51), rewords and highlights IEF287I messages.
- The second rule, called MNSTATUS (see page 61), not only suppresses IEF285I messages from displaying on the console, but also keeps them from appearing in the SYSLOG.

More information:

Using EasyRule (see page 95)

Create a Rule to Tailor the Display of Messages

To practice using EasyRule, access OPSVIEW and follow along with these steps.

To create a rule that tailors the display of messages

1. Access the EasyRule Primary panel. Begin at the OPSVIEW Primary Options Menu. Enter 2.3 into the Option field. As a result, the EasyRule Primary panel appears, a sample of which is shown here:

```
EasyRule --------------- MSI1 --- O P S V I E W --------------- Subsystem OPSS
Command ===>
        EEEEE    AAAA   SSSSS  YY   YY  RRRRR   UU  UU   LL      EEEEE
        EE      AA  AA  SS      YYYY    RR  R   UU  UU   LL      EE
        EEEE    AAAAAA  SSSSS    YY     RRRRR   UU  UU   LL      EEEE
        EE      AA  AA     SS    YY     RR RR   UU  UU   LL      EE
        EEEEE   AA  AA  SSSSS   YY     RR  RR   UUUU    LLLLL   EEEEE
ISPF LIBRARY:
   PROJECT ===>
   GROUP   ===>
   TYPE    ===>
   MEMBER  ===>
OTHER PARTITIONED DATA SET:
   DATA SET NAME  ===>
Do You Wish To AUTOMATICALLY step thru EasyRule? ===> N   (Y/N)
```

2. Specify a data set and member for the first rule. Before you can create the first rule, you must tell EasyRule the name of the rule set that will contain the rule. Use the Project, Group, and Type fields of the EasyRule Primary panel to do so. You must also specify a new member name in the Member field. Each member of a rule set contains a single rule, thus the name of the member is the name of the rule. To follow along with this example, specify these values on the EasyRule Primary panel:

   - Indicate that you want the SYS1.OPS.TEST.RULES data set to contain the new rule. This example assumes that you have already allocated a data set by this name. You can use another data set if you prefer. Type SYS1.OPS in the Project field, TEST in the Group field, and RULES in the Type field.
   - Name the rule NOTCTLG. Type NOTCTLG in the Member field.
   - Do *not* select automatic step-through for panel navigation. Instead, you will use the EasyRule menus to access the appropriate panels. Accept the default of N for the automatic step-through prompt.

Your panel should now be similar to the one shown here:

```
EasyRule --------------- MSI1 --- O P S V I E W ------------- Subsystem OPSS
Command ===>
        EEEEE    AAAA   SSSSS  YY   YY  RRRRR   UU  UU   LL      EEEEE
        EE      AA  AA  SS      YYYY    RR  R   UU  UU   LL      EE
        EEEE    AAAAAA  SSSSS    YY     RRRRR   UU  UU   LL      EEEE
        EE      AA  AA     SS    YY     RR RR   UU  UU   LL      EE
        EEEEE   AA  AA  SSSSS   YY     RR  RR   UUUU    LLLLL   EEEEE
ISPF LIBRARY:
   PROJECT ===> SYS1.OPS
   GROUP   ===> TEST
   TYPE    ===> RULES
   MEMBER  ===> NOTCTLG
OTHER PARTITIONED DATA SET:
   DATA SET NAME  ===>
Do You Wish To AUTOMATICALLY step thru EasyRule? ===> N   (Y/N)
```

After you type in the suggested field values and press Enter, the Rule Type Selection panel appears, a sample of which is shown next.

3. Select the type of rule you want to create by entering its code in the Option field of the Rule Type Selection panel.

   To follow along with this example, enter 1 to create a message rule, as shown here:

```
EasyRule ------------------------------------------------------------------
Option ===> 1
                    R U L E   T Y P E   S E L E C T I O N
     1   MSG   - Create Message Event Rule
     2   CMD   - Create Command Event Rule
     3   GLV   - Create Global Variable Event Rule
     4   TOD   - Create Time-Of-Day Event Rule
     5   OMG   - Create OMEGAMON Event Rule
     6   DOM   - Create Delete-Operator-Message Event Rule
     7   EOJ   - Create End-Of-Job Event Rule
     8   EOM   - Create End-Of-Memory Event Rule
     9   EOS   - Create End-Of-Step Event Rule
    10   TLM   - Create Time-Limit-Exceeded Event Rule
    11   USS   - Create UNIX System Services (USS) Message Event Rule

Press END to return
```

EasyRule provides a main menu for each type of rule. Since you chose option 1 on the Rule Type Selection panel to create a message rule, the Message Rule Main Menu appears next. A sample is shown next.

4. Select option 1 MESSAGE ID from the Message Rule Main Menu.

   Notice that 1 has been specified in the sample panel shown here:

```
EasyRule ------------------------------------------------------------------
Option ===> 1
             M E S S A G E  R U L E  M A I N  M E N U
  1   MESSAGE ID      -   Specify the ID of the message(s) to be processed
  2   DOCUMENTATION   -   Add comments to this Rule
  3   CONDITIONS      -   Supply additional criteria for this Rule to fire
  4   ACTIONS         -   Take action with respect to the message(s)
  5   INITIALIZATION  -   One-time initialization done when Rule is ENABLEd
  6   TERMINATION     -   Specify actions to be taken when Rule is DISABLEd
```

   In response to your selection, the Primary Event Specification Panel for message rules appears. A sample is shown next.

5. Supply the primary selection criterion. There is a unique Primary Event Specification panel for each type of rule. The primary event is the criterion that is used to execute the rule. For message rules, the primary event is always a message ID. To continue with this example, enter IEF287I in the MSG ID field.

   Notice that this has already been done in the sample panel shown here:

```
EasyRule ------------------------------------------------------------------
Command ===>
                  S P E C I F Y   M E S S A G E   I D
   MSG ID ===> IEF287I           JUST SUPPRESS ===> N  (Y/N/D)
                                        or
                                  JUST DELETE ===> N  (Y/N/D)
                               DELETE FROM OPSLOG === N  (Y/N)
   MSG ID is used to determine if this Rule should perform an Action.
   It must be 1 to 10 characters in length and may optionally include a
   "wildcard" character '*'.  MSG ID is the only required field.
   If you just want to SUPPRESS or DELETE the message, type Y next to the
   appropriate entry.  Subsequent panels are bypassed if using Step-thru mode.
   DELETE is like SUPPRESS, but also deletes the message from SYSLOG.
   D is the same as Y except that in Step-thru mode, you will be given a
   chance to enter comments about the rule
```

   After you specify the message ID for the rule, EasyRule returns you to the Message Rule Main Menu, which is shown next.

6. Select option 2 DOCUMENTATION from the Message Rule Main Menu. Notice that this sample specifies 2:

```
EasyRule ------------------------------------------------------------------
Option ===> 2
              M E S S A G E  R U L E  M A I N  M E N U
  1   MESSAGE ID      -   Specify the ID of the message(s) to be processed
  2   DOCUMENTATION   -   Add comments to this Rule
  3   CONDITIONS      -   Supply additional criteria for this Rule to fire
  4   ACTIONS         -   Take action with respect to the message(s)
  5   INITIALIZATION  -   One-time initialization done when Rule is ENABLEd
  6   TERMINATION     -   Specify actions to be taken when Rule is DISABLEd
```

In response to your selection, the Create Rule Comments panel appears. A sample panel is shown next.

7. Document the rule you are creating. It is always a good idea to provide comments for a rule. Continue with this example by entering the comments shown in the next sample of your own Create Rule Comments panel. These comments provide you with an audit trail of changes in the History section, and a general explanation of the original problem in the Rule Function section.

```
EasyRule -------------- CREATE RULE COMMENTS EXAMPLE ----------------- Tutorial
                  C R E A T E   R U L E   C O M M E N T S

 Rule Name      ===>  ARC0027I
 Rule Type      ===>  MSG
 Rule Function  ===>  When the log is switched, submit a job named "HSMLOG"
                ===>  followed by our 2-character system ID.  For example:
                ===>  HSMLOGS4 for the system we call "S4".
 Author         ===>  CA
 Support        ===>  CA
 Related Rules  ===>  ARC0026I,ARC0028I
 Related CPs    ===>  NONE
 History        ===>  ADA 2009/01/03 Original Implementation
                ===>  ADA 2009/02/01 Changed diskrdr name so it is generic
-------------------------------------------------------------------------------
  This example will generate the highlighted OPS/REXX statements:

  )MSG ARC0027I
   /*  Rule Name:      ARC0027I                                              */
   /*  Rule Type:      MSG                                                   */
   /*  Rule Function:  When the log is switched, submit a job named "HSMLOG" */

  Press ENTER to return, or END to terminate tutorial
```

After you enter comments for the rule, EasyRule returns you once more to the Message Rule Main Menu.

8. Select option 4 ACTIONS from the Message Rule Main Menu. Notice that 4 has been specified in the sample shown here:

```
EasyRule ------------------------------------------------------------------------
Option ===> 4
                M E S S A G E  R U L E  M A I N  M E N U
  1   MESSAGE ID      -   Specify the ID of the message(s) to be processed
  2   DOCUMENTATION   -   Add comments to this Rule
  3   CONDITIONS      -   Supply additional criteria for this Rule to fire
  4   ACTIONS         -   Take action with respect to the message(s)
  5   INITIALIZATION  -   One-time initialization done when Rule is ENABLEd
  6   TERMINATION     -   Specify actions to be taken when Rule is DISABLEd
```

In response to your entry, the Take Action menu for message rules appears. Use this menu to specify the actions that you want to take place when the rule is enabled. A sample Take Action menu appears next.

9. Select O, Issue Operator commands, from the Take Action menu.

This lets you issue operator commands.

For this example, suppose that you want CA OPS/MVS to write the IEF287I messages to your job log. To do this, you need to instruct CA OPS/MVS to issue two console commands. Before you can specify these commands, you must select O, for Issue Operator Commands, from the Take Action menu. Notice that in the sample panel illustrated, O has been specified in the Option field.

```
EasyRule ------------------------------------------------------------------------
Option ===>

            M E S S A G E  R U L E  --  T A K E  A C T I O N

  The actions you specify via these panels will be taken for all messages that
  have the Message ID you specified and pass any additional tests you supplied
  via the "Additional Criteria" panels.

    1  Suppress                          G  Update Global variables
    2  Message deletion                  L  Update Local or Global variables
    3  Re-route to other consoles        M  Issue z/OS messages
    4  Re-word the Message               N  Send a NetMaster Alert
    5  Hilite/Color/Change DESC codes    O  Issue Operator commands
    6  Reply (WTORs only)                P  Page support people
    7  Send to another system (MSF)      Q  Perform SQL update or insert
    8  Throttle Message display rate     S  Send messages to TSO users
    9  Update Environmental variables    U  Issue UNIX commands
                                         X  Run REXX/CLIST program in Server

Press ENTER to step thru EasyRule, or END to return
```

After you enter O in the Option field, the Issue Console Commands panel appears. A sample panel appears next.

10. Write messages to the job log. You can now enter the commands that you want CA OPS/MVS to issue when the rule is enabled. On your Issue Console Commands panel, type the commands as shown in the next example. These entries will cause CA OPS/MVS to issue two display message commands to record the problem in the job log.

    Note: These sample entries apply to a JES2 environment only. For JES3 environments, you would need to use a slightly different procedure.

    In the first command, *msg.jobnm* will be replaced by the name of the job that had the cataloging problem, followed by the text of the original IEF287I message. The second command is a warning to verify the results of the job.

```
EasyRule ------------------------------------------------------------------
Command ===>
                    I S S U E   C O N S O L E   C O M M A N D S
  CMD  1 ===> $d m {msg.jobnm},received {msg.text}__________
  CMD  2 ===> $d m {msg.jobnm},verify results_______________
  CMD  3 ===> _____________________________________________
  CMD  4 ===> _____________________________________________
  CMD  5 ===> _____________________________________________
  CMD  6 ===> _____________________________________________
  CMD  7 ===> _____________________________________________
  CMD  8 ===> _____________________________________________
  CMD  9 ===> _____________________________________________
  CMD 10 ===> _____________________________________________
  CMD 11 ===> _____________________________________________
  CMD 12 ===> _____________________________________________
  CMD 13 ===> _____________________________________________
  CMD 14 ===> _____________________________________________
  CMD 15 ===> _____________________________________________
  CMD 16 ===> _____________________________________________
```

    After you enter the console commands, EasyRule returns you to the Take Action menu. A sample is shown next.

11. Select 5 Hilite/Color/Change DESC Codes from the Take Action menu. Notice that in the following sample, 5 has been specified:

```
EasyRule ------------------------------------------------------------------
Option ===>

          M E S S A G E   R U L E  --  T A K E   A C T I O N

  The actions you specify via these panels will be taken for all messages that
  have the Message ID you specified and pass any additional tests you supplied
  via the "Additional Criteria" panels.

      1  Suppress                          G  Update Global variables
      2  Message deletion                  L  Update Local or Global variables
      3  Re-route to other consoles        M  Issue z/OS messages
      4  Re-word the Message               O  Issue Operator commands
      5  Hilite/Color/Change DESC codes    P  Page support people
      6  Reply (WTORs only)                Q  Perform SQL update or insert
      7  Send to another system (MSF)      S  Send messages to TSO users
      8  Throttle Message display rate     U  Issue UNIX commands
      9  Update Environmental variables    X  Run REXX/CLIST program in Server

Press ENTER to step thru EasyRule, or END to return
```

After you select option 5, the Hilite/Descriptor Codes panel appears. A sample is shown next.

12. Specify descriptor code as SYSFAIL. On the Hilite/Descriptor Codes panel, place the letter S in the space in front of SYSFAIL, as shown in the following example. Doing so indicates that you want to change the old descriptor code to a value of 1.

```
EasyRule ------------------------------------------------------------------
Command ===>
   M E S S A G E   R U L E  --  H I L I T E / D E S C R I P T O R   C O D E S
Use S to select one or more of the following NEW Descriptor codes:
                              S  SYSFAIL  (1)  - (Hilite, non-scrollable)
                              _  IMEDACTN (2)  - (Hilite only)
   (NOTE:  Codes 1-6 and 11   _  EVENACTN (3)
    are mutually exclusive)   _  SYSSTAT  (4)
                              _  IMEDCMD  (5)
                              _  JOBSTAT  (6)
                              _  APPLPRGM (7)
                              _  OOLMSG   (8)
                              _  OPERREQ  (9)
                              _  DYNSTAT  (10)
                              _  CRITEVET (11)
Other Descriptor code(s)                ===> __ __ __ __ __
Variable containing Descriptor code(s) ===> ____________________________
```

After you type S and press Enter, EasyRule returns you to the Take Action menu.

13. Access the EasyRule Final Options menu. From the Take Action menu, press PF3 until the EasyRule Final Options menu appears. You use the EasyRule Final Options menu to determine the disposition of the OPS/REXX code EasyRule built from your panel entries. A sample EasyRule Final Options menu is shown next.

14. Review the OPS/REXX code that EasyRule built. We recommend that you review the code EasyRule generated for your rule. To do so, choose option 3, for BROWSE, on the EasyRule Final Options menu. Notice that 3 has been specified in the sample panel shown here:

```
EasyRule --------------- MSI1 --- O P S V I E W --------------- Subsystem OPSS
Option ===>3
       EEEEE    AAAA    SSSSS  YY  YY  RRRRR  UU  UU  LL     EEEEE
       EE      AA  AA   SS      YYYY   RR  R  UU  UU  LL     EE
       EEEE    AAAAAA   SSSSS    YY    RRRRR  UU  UU  LL     EEEE
       EE      AA  AA      SS    YY    RR RR  UU  UU  LL     EE
       EEEEE   AA  AA   SSSSS   YY     RR  RR  UUUU   LLLLL  EEEEE
     1   SAVE    -  SAVE the Rule that was built and EXIT
     2   CANCEL  -  EXIT and DO NOT SAVE the Rule that was built
     3   BROWSE  -  Browse the generated OPS/REXX code
     4   ALTER   -  Return to the panels to modify the Rule
 DO YOU WANT TO BE ABLE TO MODIFY THIS RULE WITH EASYRULE?      ==> Y  (Y/N)
 DO YOU INTEND TO INSERT USER PROCESSING CODE IN THIS RULE?     ==> N  (Y/N)
 DO YOU INTEND TO INSERT USER INITIALIZATION CODE IN THIS RULE? ==> N  (Y/N)
 DO YOU INTEND TO INSERT USER TERMINATION CODE IN THIS RULE?    ==> N  (Y/N)
```

When you enter 3 on the EasyRule Final Options menu, a panel similar to the sample here appears:

```
BROWSE -- EASY_RULE_BROWSE ------------------------ Line 00000000 Col 001 080
 Command ===>                                              Scroll ===> CSR
******************************** Top of Data ********************************
)MSG IEF287I
)PROC
/*--------------------------------------------------------------------*/
/* The following code is executed each time the rule is fired.        */
/*--------------------------------------------------------------------*/
  /*------------------------------------------------------------------*/
  /* Modify the descriptor codes.  This may cause the                 */
  /* message highlighting to change on the consoles.                  */
  /*------------------------------------------------------------------*/
  MSG.DESC = OPSBITS("SYSFAIL ")
  /*------------------------------------------------------------------*/
  /* Issue MVS/JES/VM commands.                                       */
  /*------------------------------------------------------------------*/
  address OPER
    "$d m "msg.jobnm",received "msg.text
    if RC ^= 0 then
    do
      address WTO "TEXT('"||OPSINFO('PROGRAM') ,
        "issue operator command 01 failed;" ,
        "address OPER RC="||rc||"')" ,
        "MSGID('OPEASYERR')"
      do while QUEUED() ^= 0
        pull opererr
        address WTO "TEXT('"||opererr||"')" ,
        "MSGID('OPEASYERR')"
      end
    end

    "$d m "msg.jobnm",verify results"
    if RC ^= 0 then
    do
      address WTO "TEXT('"||OPSINFO('PROGRAM') ,
        "issue operator command 02 failed;" ,
        "address OPER RC="||rc||"')" ,
        "MSGID('OPEASYERR')"
      do while QUEUED() ^= 0
        pull opererr
        address WTO "TEXT('"||opererr||"')" ,
        "MSGID('OPEASYERR')"
      end
    end
  return
```

Note: The above sample is actually two separate screens. We combined them for the purpose of this example.

The panel shown on the previous page presents the OPS/REXX code EasyRule generates as a result of the panel entries suggested in this sample session. This code exists only in storage; later you will save it to the data set and member that you specified on the EasyRule Primary panel.

The following shows the OPS/REXX code and the panel entries that correspond to it:

MSG IEF287I

Rule Type Selection panel and Primary Event Specification panel

Comments box

Create Rule Comments panel

Value of MSG.DESC

Hilite/Descriptor Codes panel

Commands in the Address OPER section

Issue Console Commands panel

Return

Primary Event Specification panel

15. Save the OPS/REXX code as a new rule. When you finish browsing the generated OPS/REXX code, press PF3 to return to the EasyRule Final Options menu, shown here:

```
EasyRule --------------- MSI1 --- O P S V I E W --------------- Subsystem OPSS
Option ===> 1
       EEEEE    AAAA    SSSSS  YY  YY  RRRRR  UU  UU  LL     EEEEE
       EE      AA  AA   SS      YYYY   RR  R  UU  UU  LL     EE
       EEEE    AAAAAA   SSSSS    YY    RRRRR  UU  UU  LL     EEEE
       EE      AA  AA      SS    YY    RR RR  UU  UU  LL     EE
       EEEEE   AA  AA   SSSSS   YY     RR  RR  UUUU   LLLLL  EEEEE
     1   SAVE    -  SAVE the Rule that was built and EXIT
     2   CANCEL  -  EXIT and DO NOT SAVE the Rule that was built
     3   BROWSE  -  Browse the generated OPS/REXX code
     4   ALTER   -  Return to the panels to modify the Rule
 DO YOU WANT TO BE ABLE TO MODIFY THIS RULE WITH EASYRULE?      ==> Y  (Y/N)
 DO YOU INTEND TO INSERT USER PROCESSING CODE IN THIS RULE?     ==> N  (Y/N)
 DO YOU INTEND TO INSERT USER INITIALIZATION CODE IN THIS RULE? ==> N  (Y/N)
 DO YOU INTEND TO INSERT USER TERMINATION CODE IN THIS RULE?    ==> N  (Y/N)
```

Select 1 from the menu, as shown in the sample above. As a result, EasyRule saves the rule to the data set and member you specified earlier and returns you to the EasyRule Primary panel.

More information:

Using EasyRule (see page 95)

Create a Rule to Suppress Messages

To practice using EasyRule, access OPSVIEW and follow along with these steps to create your second rule, a message suppression rule.

To create a message suppression rule

1. Access the EasyRule Primary panel. Begin at the OPSVIEW Primary Options Menu. Enter 2.3 into the Option field. As a result, the EasyRule Primary panel appears.
2. Specify a data set and member for this second rule. You should now be on the EasyRule Primary panel. You must now tell EasyRule the name of the rule set that will contain the second rule. As you did for the first rule, use the Project, Group, and Type fields on the EasyRule Primary panel to do so. You must also specify a new member name in the Member field. Specify these values:
   - Indicate that you want the SYS1.OPS.TEST.RULES data set to contain the MNSTATUS rule. Type SYS1.OPS in the Project field, TEST in the Group field, and RULES in the Type field.
   - Name the rule MNSTATUS. Type MNSTATUS in the Member field.
   - Once again, accept the default of N for the automatic step-through prompt.

   Your panel should now look like the one shown here:

```
EasyRule --------------- MSI1 --- O P S V I E W --------------- Subsystem OPSS
Command ===>
       EEEEE    AAAA   SSSSS  YY   YY  RRRRR   UU  UU   LL      EEEEE
       EE      AA  AA  SS      YYYY    RR  R   UU  UU   LL      EE
       EEEE    AAAAAA  SSSSS    YY     RRRRR   UU  UU   LL      EEEE
       EE      AA  AA     SS    YY     RR RR   UU  UU   LL      EE
       EEEEE   AA  AA  SSSSS   YY     RR  RR   UUUU    LLLLL   EEEEE
ISPF LIBRARY:
   PROJECT ===> SYS1.OPS
   GROUP   ===> TEST
   TYPE    ===> RULES
   MEMBER  ===> MNSTATUS
OTHER PARTITIONED DATA SET:
   DATA SET NAME  ===>
Do You Wish To AUTOMATICALLY step thru EasyRule? ===> N   (Y/N)
```

   After you enter the data set and member name for the new rule, the Rule Type Selection panel appears. A sample is shown next.

3. Select the type of rule you want to create. From the Rule Type Selection panel, choose option 1 to create a message rule. Notice that this has already been done in the following sample panel:

```
EasyRule ------------------------------------------------------------------
Option ===> 1
                    R U L E  T Y P E  S E L E C T I O N
     1   MSG   - Create Message Event Rule
     2   CMD   - Create Command Event Rule
     3   GLV   - Create Global Variable Event Rule
     4   TOD   - Create Time-Of-Day Event Rule
     5   OMG   - Create OMEGAMON Event Rule
     6   DOM   - Create Delete-Operator-Message Event Rule
     7   EOJ   - Create End-Of-Job Event Rule
     8   EOM   - Create End-Of-Memory Event Rule
     9   EOS   - Create End-Of-Step Event Rule
    10   TLM   - Create Time-Limit-Exceeded Event Rule
    11   USS   - Create UNIX System Services (USS) Message Event Rule

Press END to return
```

After you select option 1, the Message Rule Main Menu appears. A sample is shown next.

4. Select MESSAGE ID from the Message Rule Main Menu. From the Message Rule Main Menu, select 1, for MESSAGE ID. Notice that 1 has been specified in the Option field in the following sample:

```
EasyRule ------------------------------------------------------------------
Option ===> 1
              M E S S A G E  R U L E  M A I N  M E N U
  1   MESSAGE ID      -   Specify the ID of the message(s) to be processed
  2   DOCUMENTATION   -   Add comments to this Rule
  3   CONDITIONS      -   Supply additional criteria for this Rule to fire
  4   ACTIONS         -   Take action with respect to the message(s)
  5   INITIALIZATION  -   One-time initialization done when Rule is ENABLEd
  6   TERMINATION     -   Specify actions to be taken when Rule is DISABLEd
```

When you choose 1 from the Message Rule Main Menu, the Primary Event Specification panel for message rules appears. A sample is shown next.

5. Supply the primary selection criterion. Use the Primary Event Specification panel to specify a primary event for the MNSTATUS rule. The primary event for a message rule is always a message ID. To continue with this example, type IEF285I in the MSG ID field. Also, specify D in the Just Delete field to indicate that you not only want to suppress the message, but you also want to keep it from appearing in SYSLOG. In the following sample, the MSG ID and Just Delete fields have been filled in:

```
EasyRule ------------------------------------------------------------------------
Command ===>
                  S P E C I F Y   M E S S A G E   I D
   MSG ID ===> IEF285I           JUST SUPPRESS ===> N  (Y/N/D)
                                        or
                                  JUST DELETE ===> D  (Y/N/D)
                              DELETE FROM OPSLOG === N  (Y/N)
   MSG ID is used to determine if this Rule should perform an Action.
   It must be 1 to 10 characters in length and may optionally include a
   "wildcard" character '*'.  MSG ID is the only required field.
   If you just want to SUPPRESS or DELETE the message, type Y next to the
   appropriate entry.  Subsequent panels are bypassed if using Step-thru mode.
   DELETE is like SUPPRESS, but also deletes the message from SYSLOG.
   D is the same as Y except that in Step-thru mode, you will be given a
   chance to enter comments about the rule.
```

When you finish specifying values on the Primary Event Specification panel and press Enter, EasyRule returns you to the Message Rule Main Menu, which is shown next.

6. Select DOCUMENTATION from the Message Rule Main Menu. From the Message Rule Main Menu, select 2, for DOCUMENTATION. Notice that this has been done in the sample shown here:

```
EasyRule ------------------------------------------------------------------------
Option ===> 2
             M E S S A G E   R U L E   M A I N   M E N U
  1   MESSAGE ID      -   Specify the ID of the message(s) to be processed
  2   DOCUMENTATION   -   Add comments to this Rule
  3   CONDITIONS      -   Supply additional criteria for this Rule to fire
  4   ACTIONS         -   Take action with respect to the message(s)
  5   INITIALIZATION  -   One-time initialization done when Rule is ENABLEd
  6   TERMINATION     -   Specify actions to be taken when Rule is DISABLEd
```

When you select 2, the Create Rule Comments panel appears. A sample Create Rule Comments panel is shown next.

7. Document the MNSTATUS rule. To continue following along with this example, type the comments that are shown in the following example onto your own Create Rule Comments panel:

```
EasyRule ------------------------------------------------------------------
Command ===>
                    C R E A T E   R U L E   C O M M E N T S
Rule Name       ===> MNSTATUS
Rule Type       ===> Message
Rule Function   ===> Monitor status is enabled to get not cat 2 msg._______
                ===> This rule deletes normal disp messages._______________
                ===> ________________________________________________________
                ===> ________________________________________________________
                ===> ________________________________________________________
                ===> ________________________________________________________
                ===> ________________________________________________________
Author          ===> CA Customer Support_____________________
Support         ===> ________________________________________________________
Related Rules   ===> NOTCTLG (IEF287I)_____________________________________
Related CPs     ===> ________________________________________________________
History         ===> 93/10/31 - Original Development_____________________
                ===> ________________________________________________________
                ===> ________________________________________________________
                ===> ________________________________________________________
```

After you enter comments for the rule, EasyRule returns you to the following Message Rule Main Menu:

```
EasyRule ------------------------------------------------------------------
Option ===>
              M E S S A G E   R U L E   M A I N   M E N U
  1   MESSAGE ID      -   Specify the ID of the message(s) to be processed
  2   DOCUMENTATION   -   Add comments to this Rule
  3   CONDITIONS      -   Supply additional criteria for this Rule to fire
  4   ACTIONS         -   Take action with respect to the message(s)
  5   INITIALIZATION  -   One-time initialization done when Rule is ENABLEd
  6   TERMINATION     -   Specify actions to be taken when Rule is DISABLEd
```

8. Access the EasyRule Final Options Menu. From the Message Rule Main Menu, press PF3 to access the EasyRule Final Options menu (shown next).

9. Review the OPS/REXX code that EasyRule built. On the EasyRule Final Options menu, choose option 3, as shown here:

```
EasyRule --------------- MSI1 --- O P S V I E W --------------- Subsystem OPSS
Option ===>3
       EEEEE    AAAA    SSSSS  YY  YY  RRRRR  UU  UU  LL     EEEEE
       EE      AA  AA   SS      YYYY   RR  R  UU  UU  LL     EE
       EEEE    AAAAAA   SSSSS    YY    RRRRR  UU  UU  LL     EEEE
       EE      AA  AA      SS    YY    RR RR  UU  UU  LL     EE
       EEEEE   AA  AA   SSSSS   YY    RR  RR  UUUU   LLLLL  EEEEE
     1   SAVE    -  SAVE the Rule that was built and EXIT
     2   CANCEL  -  EXIT and DO NOT SAVE the Rule that was built
     3   BROWSE  -  Browse the generated OPS/REXX code
     4   ALTER   -  Return to the panels to modify the Rule
 DO YOU WANT TO BE ABLE TO MODIFY THIS RULE WITH EASYRULE?      ==> Y  (Y/N)
 DO YOU INTEND TO INSERT USER PROCESSING CODE IN THIS RULE?     ==> N  (Y/N)
 DO YOU INTEND TO INSERT USER INITIALIZATION CODE IN THIS RULE? ==> N  (Y/N)
 DO YOU INTEND TO INSERT USER TERMINATION CODE IN THIS RULE?    ==> N  (Y/N)
```

When you select option 3, a panel similar to the one in the following example is displayed:

```
BROWSE -- EASY_RULE_BROWSE ------------------------ Line 00000000 Col 001 080
Command ===>                                                 Scroll ===> PAGE
******************************** Top of Data ********************************
MSG IEF285I
*********************************************************************/
*  Rule Name:     MNSTATUS                                          */
*  Rule Type:     Message                                           */
*  Rule Function: Monitor status is enabled to get not cat 2 msg.   */
*                 This rule deletes normal disp messages.           */
*  Author:        CA Customer Support                               */
*  Related Rules: NOTCLG (IEF287I)                                  */
*  History:       93/10/31 - Original Development                   */
*********************************************************************/
PROC
*-------------------------------------------------------------------*/
* The following code is executed each time the rule is fired.       */
*-------------------------------------------------------------------*/
 return "DELETE"                    /* from console and SYSLOG     */
****************************** Bottom of Data ******************************
```

This panel presents the OPS/REXX code EasyRule generates as a result of the panel entries suggested in the second part of this sample session. This code exists only in storage; later you will save it to the data set and member you indicated on the EasyRule Primary panel. The next sample panel shows the OPS/REXX code and the panel entries that correspond to it.

The CA OPS/MVS base product has the following components:

OPS/REXX Code: MSGIEF285I

Panel Entries: Rule Type Selection Panel and Specify Message ID Panel

OPS/REXX Code: Comments box

Panel Entries: Create Rule Comments Panel

OPS/REXX Code: Return DELETE

Panel Entries: Specify Message ID Panel

When you finish browsing the generated OPS/REXX code, press PF3 to return to the EasyRule Final Options menu. A sample menu is shown next.

10. Save the OPS/REXX code as a new rule. From the EasyRule Final Options menu, select 1, for SAVE. Notice that 1 appears in the following Option field:

```
EasyRule --------------- MSI1 --- O P S V I E W --------------- Subsystem OPSS
Option ===> 1
       EEEEE    AAAA    SSSSS  YY  YY  RRRRR  UU  UU  LL     EEEEE
       EE      AA  AA   SS      YYYY   RR  R  UU  UU  LL     EE
       EEEE    AAAAAA   SSSSS    YY    RRRRR  UU  UU  LL     EEEE
       EE      AA  AA      SS    YY    RR RR  UU  UU  LL     EE
       EEEEE   AA  AA   SSSSS   YY     RR  RR  UUUU   LLLLL  EEEEE
     1   SAVE    -  SAVE the Rule that was built and EXIT
     2   CANCEL  -  EXIT and DO NOT SAVE the Rule that was built
     3   BROWSE  -  Browse the generated OPS/REXX code
     4   ALTER   -  Return to the panels to modify the Rule
 DO YOU WANT TO BE ABLE TO MODIFY THIS RULE WITH EASYRULE?      ==> Y  (Y/N)
 DO YOU INTEND TO INSERT USER PROCESSING CODE IN THIS RULE?     ==> N  (Y/N)
 DO YOU INTEND TO INSERT USER INITIALIZATION CODE IN THIS RULE? ==> N  (Y/N)
 DO YOU INTEND TO INSERT USER TERMINATION CODE IN THIS RULE?    ==> N  (Y/N)
```

When you choose 1 from the EasyRule Final Options Menu, EasyRule saves the rule to the data set and member you specified earlier, and returns you to the EasyRule Primary panel.

More information:

Using EasyRule (see page 95)

Lesson 7: Suppress Messages Using the Automation Analyzer

By following the procedures outlined in this section, you can access and use the Automation Analyzer to create message suppression rules.

The Automation Analyzer is a powerful facility designed to help you analyze and modify the messages displayed by your z/OS system.

The Automation Analyzer uses the record of events stored in OPSLOG to compile a statistical analysis of the message traffic in your system. This information can help you determine which messages should be suppressed from the system console display.

Based on the information the Automation Analyzer gathers, you can ask the Analyzer to generate the rules you need or, for more elaborate automation, start EasyRule directly from the panels of the Automation Analyzer.

Prepare to Use the Automation Analyzer

Allocate a test rule set for the rules you create rather than using a production rule set.

To use the Automation Analyzer, CA OPS/MVS and OPSVIEW must be installed on your system, and you must have access to them.

For the analysis to be meaningful, sufficient time must have passed for CA OPS/MVS to collect a representative sample of events in the OPSLOG.

Gather Message Event Statistics Using Automation Analyzer

Use the Automation Analyzer to process message events only. You can use other CA OPS/MVS components o gather other types of system information.

To gather message event statistics for analysis

1. Access the OPSVIEW Primary Options Menu and enter 7.2 in the Option field.

   The Automation Analyzer Specification panel appears. A sample is shown here:

```
Automation Analyzer --- MSI1 --- O P S V I E W ---------------- Subsystem OPSS
Command ===>
Enter Automation Analyzer start and end date/time
     or leave blank to use entire OPSLOG.
                 DATE         TIME
               YYYY/MM/DD     HH:MM
     Start ===>
     End   ===>
     Use * to specify the current date or time
Analyze WTORs only?          ===> N  (Y/N)       and REPLIES? ===> N  (Y/N)
Ignore if: Command Echo?     ===> Y  (Y/N)    MPF Suppressed? ===> Y  (Y/N)
           Command Response? ===> Y  (Y/N)    Hardcopy Only?  ===> Y  (Y/N)
Use OPSLOG data from: *
    Use * to specify OPSLOG for current subsystem
```

2. Select the time range of events to be analyzed. Type start and end dates, and start and end times in the corresponding fields on the Automation Analyzer Specification panel.

   The date and time range you specify tells the Analyzer to process that portion of the OPSLOG.

3. Indicate any additional limiting criteria. In the remaining fields on the Automation Analyzer Specification Panel, indicate any additional ways in which you want to limit the analysis:

   - Restrict the analysis to only WTORs, or to only WTORs and WTOR replies.
   - Indicate whether the analysis should include echoes of commands that were issued by z/OS or other subsystems.
   - Indicate whether the analysis should include messages that MPF is suppressing.
   - Indicate whether the analysis should exclude command response messages.
   - Indicate whether the analysis should exclude messages that were issued with the hardcopy only attribute.
   - Specify the data set name of a particular OPSLOG for analysis.

4. After you complete the panel, press Enter to generate the message event statistics.

   The following Automation Analyzer Results panel appears with the generated message event statistics:

```
Automation Analyzer --- MSI1 --- O P S V I E W ------------------ ROW 1 OF 100
Command ===>                                                    SCROLL ===> PAGE
     Sel options: E - Easy Rule S - Suppress Message D - Delete Message
                  Q - Quick-Ref X - Extract Replies
     Analysis done from 2009/07/13 00:00 to 2009/07/13 23:59
     Total messages found     :     8784
     Total messages suppressed:       72 (    0.81% )
     Message       Action    # of    Percent   IBM     OPS    RuleSet  Rule
Sel  Identifier    Taken     Occr    of Total  Supp    Supp.?  Name    Name
     IST663I                 859     13.00%            0.0%  SAMPLE   SAM
     IEF196I                 815     12.33%    C       0.0%  SAMPLE   SAM
     IST530I                 393      5.94%    C       0.0%  SAMPLE   SAM
     IST314I                 329      4.97%            0.0%  SAMPLE   SAM
     IST664I                 329      4.97%            0.0%
     IST889I                 329      4.97%            0.0%
     OPS1000I                312      4.72%            0.0%
     OPC44030                196      2.96%            0.0%
     READY                   170      2.57%            0.0%
     OPS4320H                148      2.24%            0.0%
     OPS3724H                121      1.83%            0.0%
```

5. Review the Automation Analyzer Results panel. The above example panel shows a sample scrollable message event list produced by the Automation Analyzer. By default, events appear in descending order by number of occurrences. You can view different parts of the list by pressing PF7 and PF8, or by using the TOP, BOTTOM, UP *n*, and DOWN *n* commands. To change the sort order, use the SORT command. For example, issue this command to change the sort order to alphabetical order according to message ID:

   ```
   SORT MESSAGE
   ```

6. Evaluate the displayed statistics. Pay particular attention to these columns of data on the Automation Analyzer Results panel:

   Message Identifier

   Lists IDs of messages occurring in the portion of the OPSLOG you selected.

   # of Occr (Number of Occurrences)

   Shows the number of times the message ID appeared in the OPSLOG.

   Percent of Total

   Shows the percentage of total selected messages that the message ID represents.

   IBM Supp (IBM Message Suppression)

   Indicates whether the message ID appears on the IBM conservative (C), aggressive (A), or neither (blank) message suppression list.

   OPS Supp.? (CA OPS/MVS Suppressed)

   Shows the percentage of message occurrences in the selected period that was suppressed by CA OPS/MVS rules. This field should read 100 percent if the message was suppressed for the entire period.

As you evaluate the statistics, examine the messages that occur very frequently to determine whether it is important that they appear on the system console display.

For more information about the Automation Analyzer, see the *OPSVIEW User Guide*.

Suppress Messages

Use the Automation Analyzer to create message suppression rules.

After you evaluate the statistics on the Automation Analyzer Results panel, follow these steps to suppress a message.

To suppress messages

1. Find the message that you want to suppress and locate the message ID on the Automation Analyzer Results panel.
2. Type S in the Sel column next to the Message Identifier you want to suppress, and press Enter.

   The selected message will be suppressed.
3. Check the results. Make sure that the message Rule Saved appears in the Action Taken column next to the Message Identifier you selected for suppression.
4. Test and verify. Test and verify rules according to the procedures of your installation, the specific needs of your installation, or both. Use Lesson 2: Testing and Verifying a Rule as a guide.
5. Obtain approvals. Obtain installation approvals for moving the message suppression rule to a production rule set and for implementation.
6. Implement. Move the rule from the test rule set to an appropriate production rule set.

Your suppression rule is created and the messages will be suppressed.

More information

Lesson 4: How to Organize Rules into Rule Sets (see page 45)

Delete Messages

Using the Automation Analyzer, you can create a rule that not only suppresses a message, but also deletes the message from the SYSLOG.

To delete a message, enter D in the Sel column next to the corresponding message ID.

Occurrences of a message that has been deleted are still recorded in the CA OPS/MVS OPSLOG.

Access EasyRule from the Automation Analyzer

This option is meaningful only if you want to use EasyRule for something other than message suppression since the Automation Analyzer lets you generate message suppression rules directly.

To access EasyRule from the Automation Analyzer

1. Access the Automation Analyzer Results panel.
2. Enter either E or R in the Sel column.

   You are moved directly from the Automation Analyzer Results Panel into the EasyRule environment.

Lesson 8: Create Rules from an MPF Suppression List

The CA OPS/MVS MPF conversion facility lets you:

- Take advantage of the effort that your site has put into maintaining MPF message suppression lists
- Move message suppression into a more available and maintainable location
- Use libraries other than the Logical parmlib Concatenation

Use of the MPF conversion facility is limited to suppression entries. Messages regarding console colors and exit information are not automatically processed.

Prepare to Convert MPF to the Product

You must review the preparation steps listed in this topic before you can begin creating rules from an MPF suppression list.

To prepare to convert MPF suppression lists to rules

- Use ISPF option 3.2 to allocate a partitioned data set with sufficient directory space for the CA OPS/MVS rules that will be created. There will be one data set member for each MPF suppression list entry that you convert. The term *rule set* refers to a partitioned data set containing CA OPS/MVS rules.
- Use a non-production environment for testing.

  Do not use a live rule set as the initial target for your conversion. The conversion process places the source code for rules into the target data set; it is safer to use a non-production environment for testing.
- Before you can perform the conversion, CA OPS/MVS and OPSVIEW must be installed at your data center, and you must have access to them.
- If you use the MPF conversion facility, modify parmlib to remove message suppression.

Convert MPF Suppression Lists to Rules

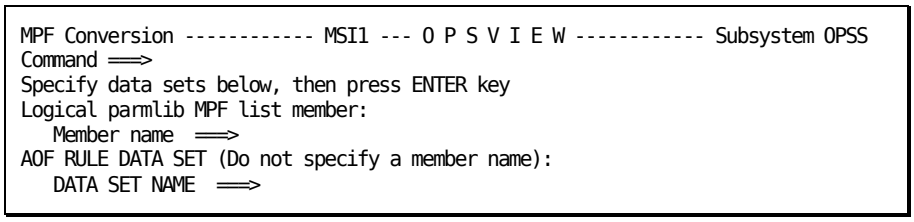
By following the procedures outlined in this section, you can:

- Access the CA OPS/MVS MPF conversion facility.
- Automatically generate CA OPS/MVS rules from an MPF suppression list.

To convert MPF suppression list to rules

1. Begin at the OPSVIEW Primary Options Menu. Enter 7.3 into the Option field.

   As a result of your selection, the MPF Conversion panel appears:

   
   ```
   MPF Conversion ------------ MSI1 --- O P S V I E W ------------ Subsystem OPSS
   Command ===>
   Specify data sets below, then press ENTER key
   Logical parmlib MPF list member:
      Member name  ===>
   AOF RULE DATA SET (Do not specify a member name):
      DATA SET NAME  ===>
   ```

2. Enter the following data in the MPF Conversion panel and press enter:

   - Type the name of the MPF list member that you want to convert in the Member name field.
   - Type the name of the test rule set (enclosed in single quotes) in the Data Set Name field.

   As a result, conversion proceeds automatically.

   Note: In recent versions of z/OS, multiple active MPF list members may exist. If this is the case at your installation, use the Member name field (as described in step 1) to enter the name of the MPF list member that you want to convert, and repeat the step as necessary. If you are using an older version of z/OS, the panel automatically displays the currently active MPF list member in the Member field.

3. Test and verify. Test and verify rules according the procedures of your installation, the specific needs of your installation, or both. Use Lesson 2 as a guideline.

   Your newly created rule is tested and verified.

4. Obtain approvals. Obtain installation approvals for canceling MPF message suppression and enabling the CA OPS/MVS rules for production.

5. Implement. Because the conversion process usually introduces CA OPS/MVS message suppression to an installation that has not used it before, you will probably need to create a new rule set to hold the generated rules.

More information:

How to Prepare Your System for Your Lessons (see page 35)
Lesson 2: Test and Verify a Rule (see page 40)

Chapter 3: Understanding CA OPS/MVS Messages

This section contains the following topics:

Message Format

CA OPS/MVS messages have the following format:

xx 0P*snnnnc*

The following explains the format of the messages.

xx

The reply number (only for WTOR, Write-To-Operator-with-Reply messages).

s

The last (fourth) character of the subsystem identifier of the copy of CA OPS/MVS. that is issuing the message. This is almost always the letter S (OPSS is the default value of the SSID parameter for the CA OPS/MVS JCL procedure member). For more information, see the *Administration Guide*.

nnnn

The message serial number.

c

The severity code for the message. The following are possible codes:

I

Informational messages. No action required.

J

Automateable informational messages. No action required. However these messages are also sent to the AOF where they are eligible to execute MSG rules.

O

Automateable messages. No action required. However these messages are also sent to the AOF where they are eligible to execute MSG rules. These messages are written to syslog and OPSLOG only.

W

Warning messages. Processing continues, but some assumptions (perhaps erroneous) are made.

E

Error messages. Some product function is lost.

S

Severe error messages. The function for an entire component is lost. These messages are non-scrollable and highlighted on the system console.

U

Unrecoverable error messages. Most, if not all, product function is lost.

A

Action messages. These messages are related to an associated operator reply message (WTOR) and provide information on the action(s) that the operator must take.

H

Hardcopy messages. These messages are written to syslog and OPSLOG only.

R

Reply messages. These messages are non-scrollable and highlighted on system consoles.

T

Trace messages. These messages are written to OPSLOG only.

Text

The text of the message. The message can contain fixed and variable data.

Note: To maintain consistency in rules you write against CA OPS/MVS messages, you must always copy and uppercase the value of the MSG.TEXT environmental variable before using it. This prevents the rules from being impacted by changes to the case of messages. For example: msgtext = TRANSLATE(MSG.TEXT).

Message Variable Data

Many messages contain variable data. The two types of variable data are local and built-in.

Local variable data is information specific to a particular message. The definitions for local variables can be found in the description of each message in the *Message Reference*.

Built-in variable data is drawn from information that is always available to messages. The following are the names and definitions of these variables:

ad

The address at which OPSNMG (the message module) was called.

The following format is used:

```
CSECTNAME+OFFSET
```

Note: This is equivalent to **cs+of**. See below.

a2

The address at which the module that called OPSNMG (the message module) was called.

The following format is used:

```
CSECTNAME+OFFSET
```

This is equivalent to **cs+o2**. See below.

cs

The CSECT name of the module calling OPSNMG.

c2

The CSECT name of the module that indirectly called module OPSNMG.

d1

The date in *YYYY/MM/DD* format.

d2

The date in *month day, year* format.

jb

The job name of the current home address space.

js

The primary Job Entry Subsystem (JES).

of

The offset in the calling module where OPSNMG was called.

o2

The offset in the callers calling module where OPSNMG was called.

pd

The product name.

ss

The subsystem name.

t2

The time in *HH.MM.SS* format.

How Messages Are Routed

The destination of a particular message and the method used to send it depend upon the environment from which the message was issued.

- Messages originating in the product main address space are issued using the WTO (Write-To-Operator) macro instruction. These messages are routed to system consoles and to the system log. They also appear in the OPSLOG Browse component of OPSVIEW. The specific consoles upon which a WTO generated message appears depends upon routing codes. These are described for the WTO macro in the IBM manual *z/OS V1R8.0 MVS Assembler Services Reference IAR-XCT*, SA22-7607-11.
- Messages issued from the TSO-based components (such as the TSO command processor, OPSCMD) are produced using the PUTLINE macro. The destination of PUTLINE output is dependent on context:
  - Interactive TMP output is routed to a TSO terminal.
  - Messages issued from within a server address space are re-sent using the WTO macro instruction.
  - Batch TMP output is routed to the SYSTSPRT ddname.

How Messages Are Captured

TSO command processor messages may be captured using one of the following methods:

- In CLISTs, messages may be captured by setting the SYSOUTTRAP CLIST variable. For more information regarding the trapping of command processor output, consult the IBM manual entitled *z/OS V1R8.0 TSO/E CLISTs*, SA22-7781-04.
- In TSO/E REXX, messages may be captured by using the OUTTRAP function. For more information regarding the trapping of command processor output, consult the IBM manual entitled *z/OS V1R8.0 TSO/E REXX Reference*, SA22-7790-07.
- In OPS/REXX programs, you can use the external data queue to trap TSO command output from commands run under the ADDRESS TSO host command environment.

Note: In OPS/REXX programs, it is generally easier and more efficient in CPU overhead to use an OPS/REXX function or host command environment instead of a TSO command processor.

AOF Processing

In general, CA OPS/MVS messages cannot be trapped by the Automated Operations Facility (AOF). However, messages that end with a severity code of **O** or **J** are exceptions to this rule. Message rules may be written for these product messages only. For information on the writing of AOF rules, consult the *AOF Rules User Guide.*

The following is a list of CA OPS/MVS messages that by default end with a severity code of **O** or **J**. This list can change with new releases of the CA OPS/MVS product, or even between releases through product fixes. Circumstances triggering a particular message and the message text itself also are subject to change. For these reasons, carefully consider the advantages and disadvantages of capturing and processing these CA OPS/MVS messages using the AOF:

- OPS0050O
- OPS0123O
- OPS0125O
- OPS1093J
- OPS1094J
- OPS1095J
- OPS1096J
- OPS1097J
- OPS2000O
- OPS2085O
- OPS3163O
- OPS3187O
- OPS3199O
- OPS3420O
- OPS3440O
- OPS3441O
- OPS3442O
- OPS3477O
- OPS3485O
- OPS3486O
- OPS3487O
- OPS3488O
- OPS3505O

- OPS3519O
- OPS3521O
- OPS3716O
- OPS3719O
- OPS3755O
- OPS3866O
- OPS3900O
- OPS3904O
- OPS4110O
- OPS4111O
- OPS4290O
- OPS4292O
- OPS4402O
- OPS4403O
- OPS5006O
- OPS7593O
- OPS7903O
- OPS7940O
- OPS7941O
- OPS7942O
- OPS8300O
- OPS8320O
- OPS8328O
- OPS8350O
- OPS8568O
- OPS8900O
- OPS8901O
- OPS8902O
- OPS8903O
- OPS9528O

Changing Message Severity Codes

The severity code of most CA OPS/MVS messages can be changed by using the OPSPARM command processor or OPSPRM OPS/REXX function.

Each message description in the *Message Reference* contains an indicator (Modifiable: Yes/No) that tells you whether the severity code of the message is eligible to be changed.

You may change message severity codes while CA OPS/MVS is initializing or after it becomes active.

Use this form of the OPSPRM OPS/REXX function to change a message severity code:

```
T = OPSPRM("SET","OPSnnnn","c")
```

Use this form of the OPSPARM command to change a message severity code:

```
OPSPARM SET(OPSnnnn) VALUE('c')
```

Reasons to change message severity codes include:

- **To suppress a message**
  - Setting the message severity code to **H** removes an unwanted message from all system consoles. The message is written only to the system log and OPSLOG.
  - Setting the message severity code to **T** removes an unwanted message from all system consoles and the system log. The message is written only to OPSLOG.
- **To reduce product initialization time**

  You can modify the startup OPS/REXX EXEC OPSSPA00 in the *hlq*.CLXSAMP library to suppress messages issued during CA OPS/MVS initialization.

  For example, you could customize the following code and place it in your OPSSPA00 EXEC:

```
/*****************************************************************
 *                                                               *
 *   Change message severity levels                              *
 *                                                               *
 *****************************************************************/
 T = OPSPRM("SET","OPS0049","T")   /* Avoid flooding consoles       */
 T = OPSPRM("SET","OPS3900","T")   /* Reduce initialization time    */
 T = OPSPRM("SET","OPS4320","T")   /* Reduce initialization time    */
```

  Once initialization is complete, you may want to change the message severity codes back to their original values.

- **To highlight a message**

  Setting the message severity code to **S** makes a message non-scrollable and highlighted on system consoles.

- **To automate a message**

  - Setting the message severity code to **J** allows the message to be automated by the AOF.
  - Setting the message severity code to **O** allows it to be automated by the AOF. The message is written only to the system log and OPSLOG.

Rules for Changing Message Severity Codes

Review the following rules before changing the severity code of any CA OPS/MVS message.

- **A severity code of T must not be changed. Doing so may result in serious problems.**
- **A severity code of R must not be changed.**
- A severity code of **I** should only be changed to severity code **J**.
- A severity code of **J** should only be changed to severity code **I**.

If none of the above rules apply, you can generally make a message available to be trapped by the AOF by changing its severity code to **J** or **O**.

View Messages Online

You can find information about CA OPS/MVS messages online using only the numeric portion of the message ID.

To view message information online

1. Use OPSVIEW option 5.5 CA OPS/MVS Message ID Lookup.
2. On the resulting display, enter the numeric portion of the CA OPS/MVS message ID you wish to review.
3. Choose the display mode (Browse or View) and press enter.

   The message information displays.

Note: For more information, see the *OPSVIEW User Guide*.

Chapter 4: Global Variables Explained

This section contains the following topics:

What Are Global Variables

The following pages explain basic and sometimes-theoretical concepts of the CA OPS/MVS global variables, as well as information that can help you use them.

Global variables are:

- OPS/REXX compound symbols
- Compound symbols that contain a defined, global stem

Features of Global Variables

CA OPS/MVS global variables have the following characteristics:

- They can contain many types of data. This data may be critical to the proper operation of CA OPS/MVS itself, and all its components, or it may be relevant only to one small, user-written application.
- They can be shared globally throughout the entire CA OPS/MVS architecture. They can be shared by, among others, TSO address spaces, AOF rules, CA OPS/MVS servers, and batch jobs.
- They can be saved across system IPLs and CA OPS/MVS restarts, as is the case with the non-volatile, standard global variables; or they can be volatile, life of the product variables that are not kept across IPLs, as is the case with temporary global variables.

Finding More Information

This table indicates where you can find more information about global variables:

Subject	Description	See...
OPS/REXX and global variables	A description of global variables in an OPS/REXX context	Global Variable Basics and OPS/REXX (see page 87) in this chapter
Global variables and CA OPS/MVS installation	Things to consider about global variables before and during the installation of CA OPS/MVS	The *Administration Guide*
Using global variables	Tips to help you use global variables	Global Variable Characteristics (see page 93) in this chapter
Global variables and OPSVIEW	An explanation of how you can manipulate global variables using OPSVIEW, the menu driven, CA OPS/MVS user interface	The *OPSVIEW User Guide*
Global variables and AOF	How to use global variables from the Automated Operations Facility perspective	The *AOF Rules User Guide*
Global variables and ESI	How you can access global variables with the Expert System Interface	Access and Update Global Variables (see page 583) in the chapter "Expert System Interface (ESI)"
Global variable backup and restore	An overview of the CA OPS/MVS feature that can backup and restore your global variable database	Backup and Restore Global Variables (see page 94) in this chapter
OPSVALUE function	Information about when you should use the OPSVALUE function	The *AOF Rules User Guide* and the *Command and Function Reference*

Global Variable Basics and OPS/REXX

This section describes global variables in the context of OPS/REXX.

To begin, the most basic definition of a global variable is that it is an OPS/REXX compound symbol that has a unique stem. An example is:

```
GLOBAL.xyz
```

To understand this definition, you must first know what an OPS/REXX compound symbol is.

OPS/REXX Compound Symbols

Compound symbols let you establish an accurate and efficient way to address values. By design, they are a good way to implement tables or arrays.

An OPS/REXX compound symbol is a named object that you can assign a value or data to. It is a powerful kind of variable because one or more variables can be included in the name of the compound symbol itself. In other words, what makes a compound symbol unique is that a portion of the name of the compound symbol itself can be a variable.

Let us back up a little. Typically, the name of the most simple type of a variable is an identifiable constant; most likely a character string.

On the other hand, consider a variable that is a compound symbol:

- It can have one or more fields of assignable, variable data in its name.
- It is identified first by the constant string portion of its name, the stem, and subsequently identified by the variable portion of its name, the tail.

Example: Compound Symbols

An analogy of this is the address system of a city.

In this analogy:

- A one dimensional, simple variable, which has only a constant string for a name, is like an address that consists only of a street name.
- A two-dimensional compound symbol, one that has a constant stem and a variable as part of the name, is like an address that includes both a street name and a number.

For more information about compound symbols, see *THE REXX LANGUAGE: A Practical* Approach to *Programming* by M.F. Cowlishaw (Prentice Hall, 1990).

Compound Symbol Format

All compound symbols have an uppercase stem that is separated from the variable name fields by a . (period or dot) character. And all the following, variable fields are also separated by a period (.) character.

Example: Format of the Compound Symbol

```
GLOBAL.a
GLOBAL.b
GLOBAL.xyz
GLOBAL.abc.xyz
```

Note: All the examples shown have GLOBAL as their stem portions. This is important later in this discussion.

Two or More Dimensions

Compound symbols can have more than two dimensions. To understand this, let us look at our city address analogy again. We have already considered the two-dimensional street address that includes a street number. In compound symbol parlance, the address 123 Main St. might look like this:

```
MAIN.123
```

What if we put in a third dimension? Consider what 123 Main St., Suite 987 would look like in compound symbol parlance:

```
MAIN.123.987
```

You could put in more dimensions; say office number 12 in Suite 987. Another dimension could be a particular in-basket in office 12. These might be represented in compound symbol parlance as:

```
MAIN.123.987.12.inbskt1
```

Compound Symbol Derived Name

To take the concept of a compound symbol to its conclusion, consider the value of its *entire name*, the stem portion, and the actual value of the variable portion. This is the derived name of a compound symbol.

Example: Illustrate the Derived Name

Suppose you have an application that has a line of code that assigns the value of 9 to a simple variable called z. And suppose that application has a subsequent line of code that initializes a compound symbol called GLOBAL.z. The derived name of GLOBAL.z, in this case, would be GLOBAL.9.

Global Compound Symbols

Now that you understand the compound symbol, consider one that can be touched by, or can touch, all components of CA OPS/MVS. Think of a compound symbol that can receive data from, or hold data for, all CA OPS/MVS components and many user-written applications. Once you have done this, you understand the CA OPS/MVS global variable.

Global Variable Nodes and Subnodes

The CA OPS/MVS terms of node and subnode are names that enable you to identify different portions of the name of a global variable and they can help you understand the relationships between global variables.

The best way to convey what nodes and subnodes represent, with respect to global variables, is by example. So consider the following:

Suppose you have the following global variables:

```
GLOBAL.A
GLOBAL.B
GLOBAL.C
GLOBAL.A.B
GLOBAL.A.B.C
GLOBAL.A.B.D
```

Let us start at the bottom and work up.

- GLOBAL.A.B.D is a subnode of GLOBAL.A.
- GLOBAL.A.B.D is an immediate subnode of GLOBAL.A.B.
- GLOBAL.A.B.C is an immediate subnode of GLOBAL.A.B.
- GLOBAL.B is a node that has no subnodes shown.

- All variables that begin with GLOBAL.A are subnodes of GLOBAL.A.
- All variables that begin with GLOBAL are subnodes of GLOBAL.

Note: In this example, GLOBAL.A.B is a totally different variable than GLOBAL.A. In fact, GLOBAL.A does not need to exist for GLOBAL.A.B to exist.

Permanent Versus Temporary Global Variables

CA OPS/MVS supports the following two basic types of global variables:

Standard or permanent

Standard global variables are non-volatile, so they are saved across system IPLs or CA OPS/MVS starts.

Temporary

Temporary global variables are not saved at these times. The set of temporary global variables is always empty when CA OPS/MVS is first started.

The following are the stems that identify these two types of global variables:

- Standard global variables
  - GLOBAL.
  - GLOBAL*x*. where *x* is any alphanumeric character (A-Z or 0-9)
- Temporary global variables
  - GLVTEMP*x*. where *x* is any alphanumeric character (A-Z or 0-9)
  - GLVEVENT
  - GLVJOBID

Temporary Global Variables: Duration Specified

CA OPS/MVS provides two temporary variable types that exist only for the duration of either an event or a job. GLVEVENT variables are associated with the duration of a specified event and GLVJOBID variables are associated with the duration of a specified job.

- GLVEVENT.stem variables are deleted when the event for which they were created terminates.

  For example, suppose that CA OPS/MVS creates GLVEVENT.MSGTEXT during the processing of an IEF405I message rule; the variable exists for all rules that process that particular IEF405I message. After the last rule associated with that individual message issuance executes, this example occurrence of GLVEVENT.MSGTEXT is automatically deleted.

  Uses for such a variable could include passing data between rules or accumulating data about a message.

- GLVJOBID.stem variables are deleted when the job or started task for which they were created terminates.

  For example, suppose that CA OPS/MVS creates GLVJOBID.MSGTEXT during the processing of a particular job; the variable exists only while that job is active. Furthermore, the variable is associated only with that job. Other jobs could have a variable by the same name containing a different value; after the last rule associated with our example job executes, this occurrence of GLVJOBID.MSGTEXT is automatically deleted.

  You can use such a variable to save data generated by one event in a job for processing in another event in that same job.

Note: As with all CA OPS/MVS temporary global variables, the number of GLVEVENT and GLVJOBID variables you have influences the amount of storage you must allocate using the GLOBALTEMPMAX parameter.

Global Variable Limits

We strongly recommend that you create no more than 10,000 global variables under a single global variable stem. The absolute product limit is 32,768 variables under a single global stem. If you do create too many variables, you will not be able to view them under OPSVIEW option 4.8 or access them using the OPSVALUE function.

Global Variable Database Warning Messages

So that you can closely monitor your global variable databases, CA OPS/MVS issues warning messages as the database becomes full. CA OPS/MVS provides parameters that enable you to closely control and monitor these database indicators. They are described in the *Administration Guide*.

CA OPS/MVS also issues warning messages each time database usage increases by 5 percent above the threshold (for instance, at 85 percent, 90 percent, and 95 percent of capacity), even between GLOBALWARNINTVAL intervals. The usage levels triggering the warning messages are not reset while CA OPS/MVS is active unless you change the GLOBALWARNTHRESH parameter to a different value. In this case, the high-usage level is reset to the threshold value.

The warning message OPS4290O, which can apply to either the permanent global variable database or the temporary global variable database, contains the following information:

- Whether the warning is for the temporary or the permanent global variable database.
- Current percentage of the database that is full.
- Number of blocks currently in use.
- Total number of blocks in the database (GLOBALMAX or GLOBALTEMPMAX).
- Name of the program or rule that, once executed, caused the threshold to be met or exceeded. This program or rule may or may not be responsible for filling the database.

Note: CA OPS/MVS checks for the threshold being exceeded only when a new global variable is allocated or an existing global variable is extended. Therefore, the interval between the messages may be greater than the defined interval.

Global Variable Characteristics

Knowing the characteristics of global variables can help you properly use CA OPS/MVS global variables.

All global variables have the following characteristics:

- They can be as long as 32,000 bytes. However, global variables containing fewer than 44 bytes operate the most efficiently and expeditiously.
- They are not declared. A global variable is created when you use a variable that contains a valid global stem.
- They can trigger global events if their values change. Changing the value of a global variable triggers a global variable event unless that variable has a stem of GLOBAL*x*. or GLVTEMP*x*. (where *x* is a number from 0 to 9).
- They are not as suited to applications that need serialization, as is the OPSVALUE function.
- The OPSVALUE function (see the *Command and Function Reference*) enables you to manipulate global variables in many ways. Because there are so many ways to use it, you must be careful that you use the correct syntax for the task that you want to accomplish. Two of the most important syntax-critical facets of OPSVALUE are as follows:
  - OPSVALUE evaluates the derived name of your global variable differently, depending on whether it is entered with quotation marks. For example:

    ```
    var = OPSVALUE('GLOBAL.A','C' ,1,2)
    var = OPSVALUE(GLOBAL.A,'C' ,1,2)
    ```

    In the first case with the single quotes, OPS/REXX evaluates the first parameter as the derived name of GLOBAL.A, whereas without quotes, it evaluates the first parameter as the actual value of GLOBAL.A
  - OPSVALUE is case sensitive when it evaluates the characters of a global variable after its stem; for example, GLOBAL.XYZ and GLOBAL.xyz are two different variables
- Global variables have derived names.
- Global variables are available to all rule-based and TSO-based OPS/REXX programs in all address spaces. You can set the value of a global variable in one rule, and then check or reset the value in another rule.

Backup and Restore Global Variables

CA OPS/MVS provides a method for you to backup and restore your global variable database.

CA OPS/MVS enables you to schedule the global variable backups or you can take a backup on demand by submitting a batch job or started task.

To schedule your global variable backups, use one of the following three methods:

- (Recommended) Internally scheduled by the CA OPS/MVS global variable checkpoint task.
- Scheduled by a CA OPS/MVS TOD rule.
- Externally controlled batch job or started task; usually controlled by a job scheduler.

For more information about the global variable backup and restore program, see the *Administration Guide*.

Chapter 5: Using EasyRule

This section contains the following topics:

EasyRule Basics

Use the CA OPS/MVS EasyRule facility to build rules without the need for programming.

EasyRule is a panel-driven rule generator, offering a fill-in-the-blanks approach to building a CA OPS/MVS rule. EasyRule not only makes generating rules fast and easy, but it also makes updating them quick and convenient.

Because you can use EasyRule to generate complex rules, even the most experienced REXX programmer will appreciate the savings in labor that EasyRule provides for producing sophisticated programming. In addition, the rules that you create with EasyRule are readily available for tailoring and maintenance.

The REXX code generated by EasyRule contains descriptive REXX comments describing the conditions and actions being performed. This makes generated REXX code easier to understand. It also makes EasyRule more useful as a way for someone unfamiliar with the REXX language to learn REXX.

Guidelines for Using EasyRule

To use EasyRule, you must be familiar with ISPF line editing and navigation, and you must understand how the console works. It is recommended that you read the *AOF Rules Guide* to familiarize yourself with rule structure.

Follow these guidelines when using EasyRule:

- You can use EasyRule to create a new rule or to modify a rule that was originally created with EasyRule. However, if you used another editing tool to create or modify a particular rule, you usually cannot use EasyRule to modify it.
- You can select the EasyRule automatic step-through feature, which takes you from one fill-in-the-blanks panel to the next without your having to make menu selections.

  Note: For more information, see Choose Automatic Versus Manual Step-through (see page 100) in this chapter.
- Any time that you want information while using EasyRule, such as an example or the definition of an unfamiliar term, press PF1/13 to access EasyRule help.
- Do *not* imbed syntax for REXX comments (for example, /* *comment-text* */) in your panel entries.

How EasyRule Builds Rules

EasyRule is comprised of numerous fill-in-the-blanks panels. As you create your rule, EasyRule keeps track of the entries you make on each panel.

Note: Your entries will include information such as events, conditions, and actions that affect the execution of the rule.

When you finish making entries on the panels, EasyRule generates the rule as OPS/REXX code (in mixed case) and retains it in memory. On the final EasyRule panel you encounter, you must choose to take one of these actions for the new rule:

- Save the rule and exit EasyRule.
- Exit EasyRule without saving the rule.
- Browse the OPS/REXX code that EasyRule generated.
- Return to the EasyRule panels to alter the rule.

You can also determine whether you want the rule to be modifiable with EasyRule, whether you intend to include user-written code in the rule, or both.

If you choose to save the rule, you can then enable, test, and disable it as you would any other rule.

Finally, you can move the rule to the production environment.

How EasyRule Benefits Different Types of Users

If you are a novice user, you can use EasyRule to generate most of the automation you need with a few panel entries.

If you are an advanced user, you can use EasyRule to generate enough OPS/REXX code to create a basic rule. Later, you can use the ISPF editing tools to add more complex logic to the rule.

The code that EasyRule generates is clean and efficient. Therefore, if you want to learn how to write OPS/REXX code, you can browse the rules EasyRule generates to learn about the OPS/REXX language.

Configure EasyRule Settings

You may want to customize your EasyRule profile. You can configure your EasyRule settings by using OPSVIEW Option 0.5, as described in the *OPSVIEW User Guide.*

Introducing EasyRule Panels

To help you to build a rule, EasyRule presents you with a series of panels that is dependent upon the type of rule you want to create. Although each set of panels is unique to the type of rule you are creating, there are similarities among them in both format and content.

For example, regardless of the type of rule you are creating, EasyRule will present a primary event specification panel on which you specify the primary criterion that is used to execute the rule. Thus, if you are creating an OMEGAMON rule, EasyRule presents you with a panel prompting you to specify an OMEGAMON exception ID; if you are creating a message rule, a panel prompting you for a message ID appears instead.

Panel Descriptions

Because of the similarities among EasyRule panels, not every EasyRule panel is presented in this chapter. Instead, EasyRule panels that are representative of the series of panels you see when you create any rule are discussed. Reviewing these pages will acquaint you with the look and feel of the EasyRule facility.

More information:

Lesson 6: Solve a Problem Using EasyRule (see page 50)

Access Additional Panel Information

If you need more specific information about how to use a particular EasyRule panel, press PF1/PF13 to access help directly from that panel.

More information:

EasyRule Help (see page 113)

How to Navigate the Panels

The typical route one takes through the EasyRule panels is as follows:

1. EasyRule primary panel
2. Rule Type Selection Panel
3. Rule Type Main Menu
4. EasyRule primary event specification panel
5. Create Rule Comments Panel
6. Conditions Menu
7. Take Action Menu
8. Initialize Rule Variables Panel
9. Actions to Take at Rule Disable Panel
10. EasyRule final options menu

Note: If you select the EasyRule automatic step-through feature, you bypass all EasyRule selection menus.

More information:

Choose Automatic Versus Manual Step-through (see page 100)

How to Access EasyRule

You can use these methods to access EasyRule:

- From the OPSVIEW Primary Options Menu, type 2.3 and press Enter.
- From the AOF TEST Rule List panel (OPSVIEW option 2.1) or the AOF CTRL Rule List panel (OPSVIEW option 4.5.1), enter the rule set name and press Enter. Do either of the following:
  - Type R next to the name of the rule you want to modify and press Enter.
  - Enter the EASYRULE primary command in the Command field to create a new rule.
- From the Automation Analyzer Results panel (OPSVIEW option 7.2), type E or R next to the message ID for which you want to create or modify a rule, and then press Enter.

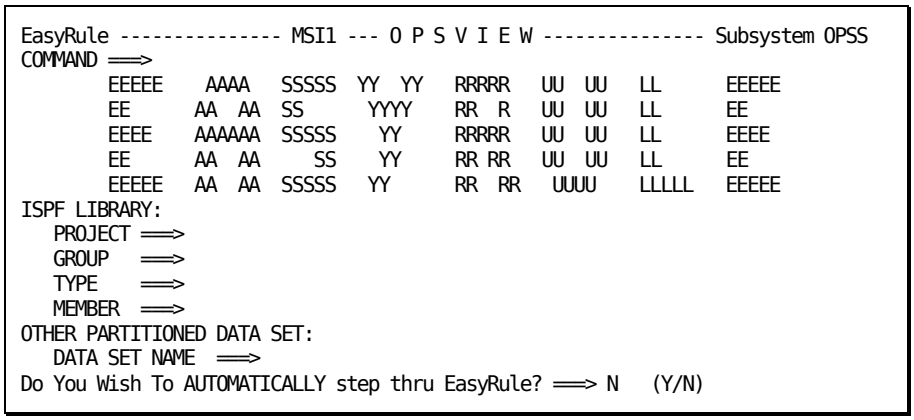
Access EasyRule from OPSVIEW and Specify a Rule Set

Before you can create or modify a rule, you must tell EasyRule the name of the rule set that will contain (or already contains) the rule.

To access EasyRule from OPSVIEW and Specify a Rule Set

1. From the OPSVIEW Primary Options Menu, enter 2.3 into the Option field and press Enter.

   As a result, the EasyRule primary panel appears. A sample panel is shown here:



2. Complete the Project, Group, and Type fields.
3. Specify a new or existing member name in the Member field. Each member of a rule set contains a single rule.

Access EasyRule from the AOF Test or AOF Control Facility

You can access EasyRule in three ways:

To modify an existing rule:

1. From the AOF TEST Rule List panel (OPSVIEW option 2.1) or the AOF CTRL Rule List panel (OPSVIEW option 4.5.1), enter the rule set name and press Enter.
2. Type R next to the name of the rule you want to modify and press Enter.

   In response, EasyRule takes you directly to the main menu for the type of rule you selected.

To create a new rule:

1. From the AOF TEST Rule List panel (OPSVIEW option 2.1) or the AOF CTRL Rule List panel (OPSVIEW option 4.5.1), enter the rule set name and press Enter.
2. Enter the EASYRULE primary command in the Command field.

   In response, the EasyRule primary panel appears displays.

To access EasyRule from the Automation Analyzer:

1. Access the Automation Analyzer Results panel (OPSVIEW option 7.2)
2. Type R or E next to the message ID for which you want to create or modify a rule; then press Enter.

   In response, the EasyRule primary panel appears.

Choose Automatic Versus Manual Step-through

When you access EasyRule, you must decide how you want to proceed: manually or automatically.

To choose either automatic or manual step-through

When the EasyRule primary panel prompts you for your choice:

- If you want to move through the EasyRule panels by making menu selections, type N in response to the prompt.
- If you want EasyRule to move you from one fill-in-the-blanks panel to the next, without presenting you with menus, type Y in response to the prompt.

Scrollable Menu and Data Entry Panels

Many EasyRule menu and data entry panels are scrollable so that more information can be included in the panel than will fit on a single terminal screen. The OPSINFO Data Conditions panel, for example, includes many more choices than can fit on a single display screen. When scrollable text is present, the word More: appears near the upper-right corner of your screen. Use the PF7 and PF8 keys to scroll up and down, respectively.

PF10 and PF11 are used to control scrolling in Help panels.

Select a Rule Type

EasyRule lets you create many different types of rules. The Rule Type Selection panel lets you easily choose the type of rule to create.

To select a rule type

1. If you specified the member name for a new rule on the EasyRule primary panel, the Rule Type Selection panel appears. EasyRule provides a main menu for each type of rule.

```
EasyRule ------------------------------------------------------------------
OPTION ===>
                    R U L E   T Y P E   S E L E C T I O N
     1   MSG   - Create Message Event Rule
     2   CMD   - Create Command Event Rule
     3   GLV   - Create Global Variable Event Rule
     4   TOD   - Create Time-Of-Day Event Rule
     5   OMG   - Create OMEGAMON Event Rule
     6   DOM   - Create Delete-Operator-Message Event Rule
     7   EOJ   - Create End-of-Job Event Rule
     8   EOM   - Create End-Of-Memory Event Rule
     9   EOS   - Create End-Of-Step Event Rule
    10   TLM   - Create Time-Limit-Exceeded Event Rule
    11   USS   - Create Unix Systems Services (USS) Message Event Rule
```

   If you specified an *existing* rule on the EasyRule primary panel, EasyRule bypasses the panel shown above and takes you directly to the series of panels you can use to modify the rule.

2. Enter the code into the Option field. For example, if you want to create a message rule, enter 1 in the Rule Type Selection panel.

   The Message Rule Main Menu appears. A sample is shown here:

```
EasyRule ------------------------------------------------------------------
OPTION ===>
              M E S S A G E   R U L E   M A I N   M E N U
  1   MESSAGE ID      -   Specify the ID of the message(s) to be processed
  2   DOCUMENTATION   -   Add comments to this Rule
  3   CONDITIONS      -   Supply additional criteria for this Rule to fire
  4   ACTIONS         -   Take action with respect to the message(s)
  5   INITIALIZATION  -   One-time initialization done when Rule is ENABLEd
  6   TERMINATION     -   Specify actions to be taken when Rule is DISABLEd
```

   The rule type of Message Rule is selected and you are ready to specify a primary event for your rule.

Rule Type Main Menu Options

Although there is a unique Rule Type Main Menu for every type of rule you can create with EasyRule, all the panels offer similar options:

Option 1 MESSAGE ID

Accesses a panel on which you specify the primary selection criterion for this type of rule. For example, if you are creating a command rule, specify the command; if you are creating an OMEGAMON rule, specify the exception ID.

Option 2 DOCUMENTATION

Accesses a panel on which you enter comments for the rule. EasyRule incorporates your comments into the OPS/REXX code it generates for the rule. This panel is the same regardless of what type of rule you are creating.

Option 3 CONDITIONS

Accesses a sub-menu from which you can choose conditions for EasyRule to use as selection criteria for the rule. The content of this sub-menu varies for different types of rules.

Note: These conditions *include* the primary selection criterion you specify with option 1.

Option 4 ACTIONS

Accesses a sub-menu from which you specify the actions the rule should take when the conditions set in option 3 are met (if any). If no conditions are set in option 3, the actions you specify in option 4 will occur unconditionally. The content of this sub-menu is different for each type of rule.

Option 5 INITIALIZATION

Accesses a panel on which you specify the initial values of local and global variables. EasyRule performs the initialization when the rule is first enabled. This panel is the same for all types of rules.

Option 6 TERMINATION

Accesses a panel on which you specify the actions CA OPS/MVS should take when the rule is disabled. Possible actions include sending messages, setting global variables, and so on. This panel is the same for all types of rules.

Specify a Primary Event for a Rule

The primary event is the criterion that is used to execute the rule. For message rules, the primary event is a message ID; for OMEGAMON rules, it is an OMEGAMON exception ID, and so on.

To specify a primary event for a rule

1. From the Message Rule Main Menu, choose option 1 Message ID to access the primary event specification panel to specify a primary event for your rule.

   Following is a sample primary event specification panel pertaining to message rules:

```
EasyRule ------------------------------------------------------------------------
Command ===>

                  S P E C I F Y   M E S S A G E   I D

   MSG ID ===> IEF287I            JUST SUPPRESS ===> N  (Y/N/D)
                                       or
                                  JUST DELETE ===> N  (Y/N/D)

                              DELETE FROM OPSLOG === N  (Y/N)

   MSG ID is used to determine if this Rule should perform an Action.
   It must be 1 to 10 characters in length and may optionally include a
   "wildcard" character '*'.  MSG ID is the only required field.

   If you just want to SUPPRESS or DELETE the message, type Y next to the
   appropriate entry.  Subsequent panels are bypassed if using Step-thru mode.
   DELETE is like SUPPRESS, but also deletes the message from SYSLOG.

   D is the same as Y except that in Step-thru mode, you will be given a
   chance to enter comments about the rule.

Press ENTER to step thru EasyRule, or END to return
```

   When EasyRule generates the OPS/REXX code for your rule, the type of the rule and the primary event of the rule make up the first line of the code.

   In some cases, the only rule type-specific panel that you need to complete to create a rule is the primary event specification panel. For example, if your goal is simply to suppress a message, you can do so by making only two field entries (MSG ID and Just Suppress) on the primary event specification panel for message rules.

2. You can then press PF3

   The EasyRule final options menu displays, from which you can end your EasyRule session.

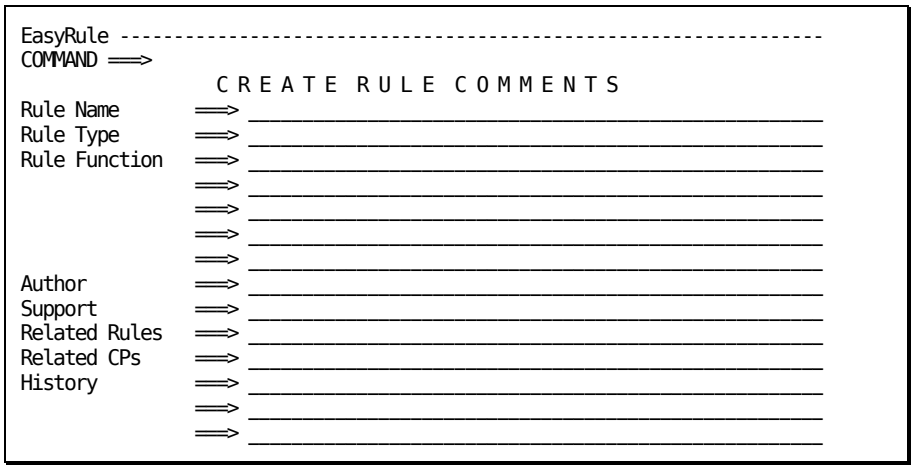
Specify Comments for a Rule

The Create Rule Comments panel provides a structured format that you can use to create useful documentation for your rule.

To specify comments for a rule

1. Access the Create Rule Comments panel.

   A sample panel, the same for all types of rules, follows:



2. Enter comments about the rule.

No editing is performed on your entries, all of which are optional. EasyRule takes your entries and generates valid REXX comment lines from them. Do not imbed syntax for REXX comments (for example, /* comment-text */) in your entries.

Specify Execution Conditions for a Rule

Use the Conditions Menu to choose conditions for EasyRule to use as selection criteria for the rule. These conditions are in addition to the primary selection criterion you specified on the primary event specification panel. A unique Conditions Menu exists for each rule type. A sample panel, showing the Conditions Menu for message rules, appears here.

Note: If you select the EasyRule automatic step-through feature, you bypass the Conditions Menu (and all other EasyRule menus).

```
EasyRule ------------------------------------------------------------------
Option ===>

       M E S S A G E   R U L E  --  C O N D I T I O N S   M E N U

   These panels allow you to specify additional criteria (beyond the Message
   ID) which must be satisfied for this Rule to fire.

   1  Message text                          A  Other address spaces
   2  Current console routing (ROUTCDEs)    D  Day/Time/Shift/Calendar
   3  Highlighting/Color (DESC codes)       G  Global variables
   4  Message Environmental variables       L  Local or other Global variables
   5  Multi-line WTO support                O  OPSINFO variables
                                            V  Device or VOLSER status

   C  Specify how multiple conditions are to be evaluated (AND/OR)

Press ENTER to step thru EasyRule, or END to return
```

Although the total number of options on the menus differs among rule types, the format of the menus is the same. The column on the left of each Conditions Menu lists conditions that are unique to the type of rule you are creating. The column on the right lists conditions that are common to all rule types.

To choose an option from a Conditions Menu

1. Type its option code into the Option field and press Enter.
2. When you complete the resulting panel, press PF3

   You are returned to the Conditions Menu.

You can repeat this procedure to choose several or all options, as long as you do so one at a time.

Specify Actions to Be Taken When a Rule Executes

Use the Take Action menu to specify the actions that you want to occur when the rule executes and all criteria are satisfied. A unique Take Action menu exists for each rule type. The following sample panel shows the Take Action menu for message rules.

Note: If you select the EasyRule automatic step-through feature, you bypass the Take Action menu (and all other EasyRule menus).

```
EasyRule ------------------------------------------------------------------
Option ===>

            M E S S A G E   R U L E  --  T A K E   A C T I O N

  The actions you specify via these panels will be taken for all messages that
  have the Message ID you specified and pass any additional tests you supplied
  via the "Additional Criteria" panels.

      1  Suppress                          G  Update Global variables
      2  Message deletion                  L  Update Local or Global variables
      3  Re-route to other consoles        M  Issue z/OS messages
      4  Re-word the Message               N  Send a NetMaster Alert
      5  Hilite/Color/Change DESC codes    O  Issue Operator commands
      6  Reply (WTORs only)                P  Page support people
      7  Send to another system (MSF)      Q  Perform SQL update or insert
      8  Throttle Message display rate     S  Send messages to TSO users
      9  Update Environmental variables    U  Issue UNIX commands
                                           X  Run REXX/CLIST program in Server
Press ENTER to step thru EasyRule, or END to return
```

Although the total number of options on the menus differs among rule types, the format of the menus is the same. The column on the left of each Take Action Menu lists actions that are unique to the type of rule you are creating. The column on the right lists actions that are common to all rule types.

To choose an option from a Take Action menu

1. From the Message Rule Main Menu, choose option 4 Actions.

   The Message Rule Take Action menu displays.

2. When you complete the resulting panel, press PF3.

   You are returned to the Take Action menu.

You can repeat this procedure to choose several or all options, as long as you do so one at a time.

Specify Actions to Be Taken When a Rule Is Enabled

Use the Initialization menu to choose the actions that you want to perform during enable processing.

To specify actions to be taken when a rule is enabled

1. From the Message Rule Main Menu, choose option 5 Initialization.

   The Initialization menu displays.

2. Choose the actions you want to perform during enable processing and press Enter.

   While a rule is enabled, it executes when its primary criterion is met, such as when a particular message is issued by the system, or when a particular job ends. A rule that is not enabled never executes.

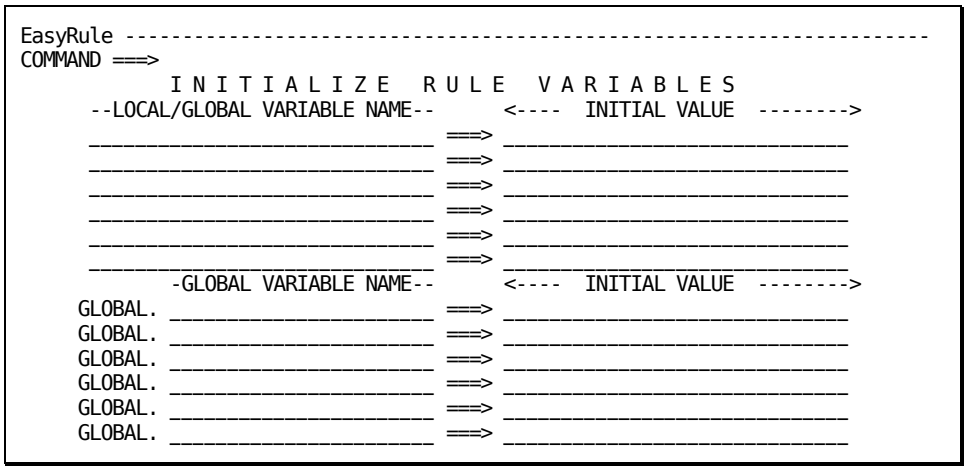
   EasyRule provides two ways to specify a limited amount of processing during enable processing:

   - You can set the value of a variable.
   - You can elect whether to allow the rule to enable.

If any variables are set during enable processing, they will remain set even if the rule ultimately rejects the enable request.

Set Initial Variable Values for a Rule

Use the Initialize Rule Variables panel to set variables to specific values before the rule executes for the first time. The following sample panel applies to all types of rules:



To set initialize variable settings for a rule

1. Type the name of the variable in the left-hand column.
2. Type the desired initial value in the right-hand column.

The entries you make on this panel become OPS/REXX statements in the)INIT section of the code EasyRule generates for your rule. These OPS/REXX statements execute only when the rule is first enabled.

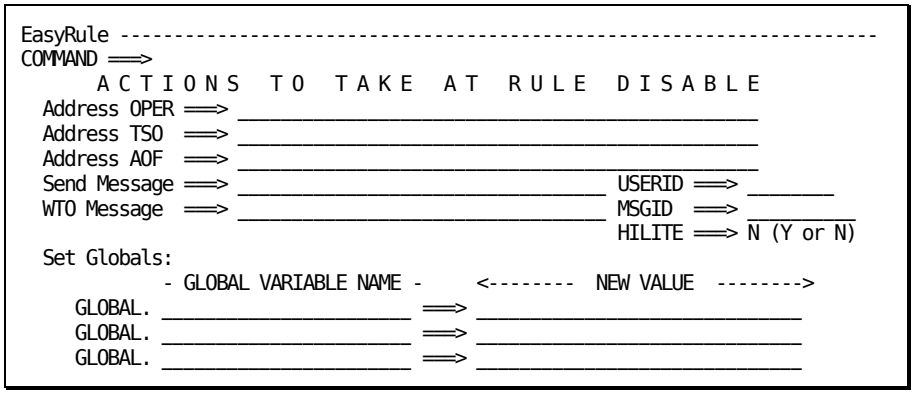
Specify Actions to Be Taken When a Rule Is Disabled

Use this panel to specify the actions that CA OPS/MVS should take when the rule is disabled.

To specify the actions CA OPS/MVS should take when a rule is disabled

1. From the rule type main menu, choose Option 6: Termination - Specify actions to be taken when Rule is DISABLEd.

   The following Actions to Take at Rule Disable menu displays, which applies to all types of rules:



2. Complete the fields for the action you want. For example, you may want to send a message, trigger some action on the system, or set a global variable to a new value.

   The entries you make on this panel become OPS/REXX statements in the)TERM section of the code EasyRule generates for your rule. These OPS/REXX statements execute only when the rule is disabled.

EasyRule Final Options Menu Determines the Disposition of a Rule

After you complete all of the necessary EasyRule panels, the EasyRule final options menu appears. You use this panel to determine the disposition of the OPS/REXX code EasyRule built from your panel entries. The following sample panel applies to all types of rules:

```
EasyRule --------------- XE44 --- O P S V I E W --------------- Subsystem OPSA
Option ===>


        EEEEE    AAAA    SSSSS  YY  YY  RRRRR  UU  UU  LL     EEEEE
        EE      AA  AA   SS      YYYY   RR  R  UU  UU  LL     EE
        EEEE    AAAAAA   SSSSS    YY    RRRRR  UU  UU  LL     EEEE
        EE      AA  AA      SS    YY    RR RR  UU  UU  LL     EE
        EEEEE   AA  AA   SSSSS    YY    RR  RR  UUUU   LLLLL  EEEEE


      1   SAVE    -  SAVE the Rule that was built and EXIT
      2   CANCEL  -  EXIT and DO NOT SAVE the Rule that was built
      3   BROWSE  -  Browse the generated OPS/REXX code
      4   ALTER   -  Return to the panels to modify the Rule


  DO YOU WANT TO BE ABLE TO MODIFY THIS RULE WITH EASYRULE?      ==> Y  (Y/N)
  DO YOU INTEND TO INSERT USER PROCESSING CODE IN THIS RULE?     ==> Y  (Y/N)
  DO YOU INTEND TO INSERT USER INITIALIZATION CODE IN THIS RULE? ==> Y  (Y/N)
  DO YOU INTEND TO INSERT USER TERMINATION CODE IN THIS RULE?    ==> Y  (Y/N)

Press END to SAVE and return
```

Choose one of these options on the EasyRule final options menu:

Option 1 - SAVE

EasyRule saves the rule into the member you specified on the EasyRule primary panel.

Note: If you prefer, you can press PF3 to achieve the same result.

Option 2 - CANCEL

EasyRule ignores all of your panel entries. No rule is created or updated.

Note: If you prefer, you can enter the CANCEL command into the Option field of the panel to achieve the same result.

Option 3 - BROWSE

EasyRule takes you to the standard ISPF browse panel, where you can review the OPS/REXX code that it built for your rule. You cannot make changes to the OPS/REXX code from the ISPF browse panel. If you want to make changes, press PF3 to return to the EasyRule final options menu, and then choose option 4 (ALTER).

Option 4 - ALTER

EasyRule returns you to the primary event specification panel and moves you through all of the subsequent panels. The values that you entered appear on the panels. You can make changes, additions, or deletions to the specifications of the rule as you view your entries. When you finish, press PF3 to return to the EasyRule final options menu.

Regardless of which option you choose, an ISPF message appears in the upper-right corner of the panel to indicate whether the rule has been saved.

Make Modifications with EasyRule

The EasyRule final options menu prompts you to decide whether you want EasyRule to be the mechanism with which you make further changes to the rule.

The default is Y. If you enter N, you will *not* be able to use EasyRule to modify the code in the future.

Work With User Code Areas in EasyRule

Sometimes you might want to write a rule in which decisions are made or actions taken that cannot be achieved with EasyRule. For example, you might want to test system status using a complex set of AND/OR logic equations, or you might want to choose one of several actions depending on system status or other criteria. EasyRule provides optional user code areas, which are reserved spaces in a rule that was generated by EasyRule. In these user code areas you can add your own REXX statements to the automatically generated ones, so you can extend EasyRule-generated rules to support decisions and actions of any degree of complexity. You can use ISPF edit to add or change REXX code in a user code area. Code generated by EasyRule and saved as an EasyRule-modifiable rule should *only* be changed by using EasyRule.

The EasyRule save panel provides three user code area controls:

- Initialization
- Termination
- Processing

Your responses to the three questions on this panel determine whether the respective user code areas will be generated for that rule. You can also allow all three responses to default to the OPSVIEW 0.5 value (for details on using this option, see the *OPSVIEW User Guide*).

How to Test a Rule

After you create a rule, use the AOF test facility to test it before you put it into production.

EasyRule Error Messages

Error messages generated by EasyRule have a message ID of OPEASYERR.

EasyRule Help

EasyRule help is designed to provide detailed information about each EasyRule panel at the touch of a key. Through EasyRule help, you can learn:

- What a panel does
- What the valid input values are for a panel
- What OPS/REXX code EasyRule will generate as a result of typical panel entries
- What an unfamiliar term means

From any EasyRule panel, take either of these actions to access help:

- Press PF1/13.
- Enter the HELP command in the Command or Option field.

Basic Types of EasyRule Help Panels

EasyRule provides the following types of help panels:

- Menu help panels
- Standard help panels
- Help example panels
- Help glossary panels

Access and Use the Menu Help Panels

Menu help panels provide a little more detail about each of the available menu options. If you access help from an EasyRule menu panel, you access the help panel for that menu.

To access and use the menu help panels

1. Press PF1/13 from any EasyRule panel, in this example from the Conditions Menu for message rules.

   The following menu help panel appears:

```
EasyRule --------------- MESSAGE RULE -- CONDITIONS MENU ------------- Tutorial
Option ===>

MENU INSTRUCTIONS:   Type the menu selection number in the Option ===> field.

These options look for:

1 - MESSAGE TEXT            - Generic or specific message words
2 - CURRENT CONSOLE ROUTING - Route Code(s)
3 - HIGHLIGHTING/COLOR      - Descriptor Code(s)
4 - ENVIRONMENTAL VARIABLES - Generically or exactly specified values
5 - MULTI-LINE WTO          - Test for multi-line or single line WTO

These options make use of:

A - OTHER ADDRESS SPACES    - Determine whether other address spaces are active
D - DAY/TIME/SHIFT/CALENDAR - Include or exclude by time/day conditions
G - GLOBAL VARIABLES        - Exact or generic global variable values
L - LOCAL/GLOBAL VARIABLES  - Exact or generic local or global variable values
O - OPSINFO VARIABLES       - Values about the product and/or its environment
V - OPSDEV STATUS           - ONLINE/OFFLINE status of a device or VOLSER

Type selection number for detailed tutorial, or press END to terminate
```

2. You may take these actions on menu help panels:

   - Enter a selection number or letter in the Option field.

     A more detailed help panel for the selected option appears.

   - Press PF3.

     The EasyRule help session is terminated.

Access and Use the Standard Help Panels

Another type of help panel you can access from EasyRule is a standard help panel.

To access and use the standard help panels

1. Press PF1/13 from the Message Rule-Descriptor Code Conditions panel.

   The following standard help panel appears:

```
EasyRule ------- MESSAGE RULE - HILITE/DESCRIPTOR CODES ---------- Tutorial
Option ===>

  PURPOSE:  To assign new Descriptor Codes to this Message Rule.

   HOW TO:  Type an S next to each character string whose Descriptor Code
            you wish applied to this Rule.

            Type in up to five additional Descriptor Codes from 12 to 16
            to apply to this Rule OR type in a variable name that contains
            the desired Descriptor Codes.

 POSSIBLE   See the Glossary for an explanation of Descriptor Codes.
   INPUTS:  Type V for an explanation of valid variable syntax.

            Codes 1-6 and 11 are mutually exclusive.  However, codes
            7 through 10 can be assigned in combination with other codes.

  RESULTS:  Descriptor Codes selected on this panel will be placed as
            assignments in the )PROC section of the Rule's generated
            OPS/REXX code.

Type E for Example, G for Glossary, or press END to terminate tutorial
```

   Standard help panels group information into these sections:

   PURPOSE

   Explains the purpose of the panel.

   HOW TO

   Provides basic instructions for the entries you need to make on the panel. Tells you whether an entry is required or optional.

   POSSIBLE INPUTS

   Explains what types of entries are valid for a particular field. On some panels, provides further details for particular fields.

   RESULTS

   Describes the OPS/REXX code that EasyRule will generate as a result of your entries.

2. You may take these actions on most standard help panels:

   - Enter E in Option field.

     A help example panel appears. It presents you with both an example of a correctly filled-in panel, and the OPS/REXX code that EasyRule would generate from those entries.

   - Enter G in Option field.

     A help glossary panel appears. In the EasyRule glossary, you can look up the definitions of unfamiliar terms.

   - Press Enter.

     You are presented with a second standard help panel.

     Note: This applies to two-part standard help panels only.

   - Press PF3.

     Your EasyRule help session is terminated.

Access and Use the Help Example Panels

On most of the EasyRule standard help panels, you can see an example of a correctly filled-in panel, along with the OPS/REXX code that EasyRule would generate from those entries.

To access and use the help example panels

1. Enter E on a standard help panel.

   The example of a correctly filled-in help example panel appears:

```
EasyRule ----- MESSAGE RULE - DESCRIPTOR CODE CONDITIONS EXAMPLE ----- Tutorial
COMMAND ===>
M E S S A G E   R U L E  -  D E S C R I P T O R   C O D E   C O N D I T I O N S
Use S to select one or more of the following Descriptor Codes:
                              S  SYSFAIL  (1)  (Hilite, non-scrollable)
                                 EVENACTN (2)  (Hilite only)
                                  ....
                              S  DYNSTAT  (10)
                                 CRITEVET (11)
Other Descriptor Code(s)              ===>
-------------------------------------------------------------------------------
  This example will generate the highlighted OPS/REXX statements:
  )PROC
   EASYRULEDESC = OPSBITS("SYSFAIL")
   EASYRULEDESC = BITOR(EASYRULEDESC,OPSBITS("DYNSTAT"))
   IF  (BITAND(MSG.DESC,EASYRULEDESC) = BITOR(EASYRULEDESC,"0000"X))
     THEN DO ...
```

   Help example panels group information into two sections:

   - The top half of the panel shows the EasyRule panel of which you wanted an example. The panel is filled in with a typical set of entries, which are highlighted.
   - The bottom half of the panel shows the OPS/REXX code that EasyRule would generate for the entries in the top half. If necessary, the bottom half also describes the entries.

   Note: Sometimes, due to space limitations, not all lines of generated code appear.

2. Take these actions on help example panels:

   - Press Enter.

     Returns you to the previous help panel.

   - Press PF3.

     Terminates your EasyRule help session.

The help example panel has been accessed and reviewed.

Access and Use Help Glossary Panels

On most of the EasyRule standard help panels, you can enter G in the Option field to access the EasyRule glossary.

To access and use the EasyRule glossary

1. Enter G in the Option field to access the EasyRule glossary.

   The following sample help glossary displays:

```
EasyRule ----------------------- GLOSSARY -------------------------- Tutorial
COMMAND ===>
   DESCRIPTOR CODES:  Two-digit values that indicate the means of message
                      presentation and message deletion on display devices.
                      1 System Failure                 8 Out-of-Line Message
                      2 Immediate Action Required      9 Operator Request
                      3 Eventual Action Required      10 Dynamic Status Display
                      4 System Status                 11 Critical eventual
                      5 Immediate Command Response       action requested
                      6 Job Status                   12-16 Reserved for
                      7 Application program/processor;      future use
                        message is to be deleted when
                        issuing task is terminated
                       Codes 1-6 and 11 are mutually exclusive.  Codes 7-10
                       can be assigned in combination with any other code.
```

   Help glossary panels are arranged in alphabetical order. Typically, when you access the glossary, you are not positioned at its beginning.

2. Take these actions on help glossary panels:

   - Enter B in Command field.

     Moves you to the TOP of the glossary, or back to the previous help panel.

   - Press Enter.

     Moves you forward in the glossary.

   - Press PF3

     Terminates your EasyRule help session.

Chapter 6: Using OPS/REXX

This section contains the following topics:

OPS/REXX Overview

A crucial part of the CA OPS/MVS product, OPS/REXX is a powerful, SAA-compliant programming language that adds to standard REXX a set of extensions that automate and enhance the productivity of z/OS operations.

Because OPS/REXX differs only slightly from standard REXX, this chapter explains the differences between OPS/REXX and standard REXX instead of describing the REXX language completely.

Additional OPS/REXX and standard REXX documentation:

- For a list of OPS/REXX built-in functions that differ from and extend standard REXX, see the chapter “OPS/REXX Built-in Functions” in the *Command and Function Reference*.
- For more detailed information about standard REXX, consult the book *THE REXX LANGUAGE: A Practical Approach to Programming* by M.F. Cowlishaw. If you do not have a copy of this book, you can order one from Prentice Hall.

  Important! If you are not already familiar with REXX, see the Cowlishaw book before you read further in this chapter.
- You also may want to read *Modern Programming Using Rexx* by Robert P. O'Hara and David Roos Gomberg (Prentice Hall). This book includes many practical examples of REXX programming.

Why OPS/REXX

We chose REXX as the programming language for CA OPS/MVS because it is the most powerful and easiest-to-use command language available. OPS/REXX provides a simple but capable high-level language to write operating system exits.

OPS/REXX Performs Better

The OPS/REXX interpreter runs many times faster than the TSO EXEC command for similar programs. Also, when you use OPS/REXX in the AOF environment, all the code is pre-interpreted to speed processing even further. OPS/REXX generally runs so speedily that most users do not need to rewrite functions in assembler language.

OPS/REXX provides automation from the point where it intercepts messages and commands from JES2 or JES3 and z/OS. Approaches that try to use TSO/E REXX or NetView for automation are more limited; they can only compare command and message events against categories in an event table and execute a program if the event matches one of those categories.

OPS/REXX Is Easy to Learn

If you can program in *any* language, you can learn to program in OPS/REXX. All variables are treated as character strings, which OPS/REXX can convert to numeric values and reconvert to character strings automatically as required.

Powerful Data Handling Tools

OPS/REXX provides facilities for writing structured programs. It supports all common program structures such as DO WHILE ... END and IF ... THEN ... ELSE. OPS/REXX also provides many built-in functions to handle dates and times, and to convert binary and hexadecimal data to or from decimal or character formats.

OPS/REXX also supports subroutine and function calls to and from other languages, as well as to and from other OPS/REXX programs.

Understandable Error Messages

In developing standard REXX, IBM designed understandable error messages. Because OPS/REXX is so similar to standard REXX, we have adopted these well-thought-out error messages for our product. The Cowlishaw book describes messages with error codes up to 49. The CA OPS/MVS online message documentation describes errors with higher codes.

Uses of OPS/REXX in the Product

The following CA OPS/MVS components use OPS/REXX:

- The rules of the CA OPS/MVS AOF component are actually OPS/REXX programs that can respond automatically to system events. The availability of the OPS/REXX general purpose programming tools in rules gives you an unlimited ability to automate responses to these events.
- Important parts of OPSVIEW such as AOF EDIT, the ISPF Dialog Manager application with which you create and update rules, are written in OPS/REXX. The OPS/REXX interface to the Dialog Manager is as complete and powerful as that of TSO/E REXX.
- You can write AOF asynchronous procedures (which execute in CA OPS/MVS server address spaces) in both OPS/REXX and the TSO/E REXX language.
- OPS/REXX differs from other REXX implementations in processing numeric values and implementing NUMERIC DIGITS. OPS/REXX attempts to optimize numeric processing for values that fit in a 32 bit fullword as a signed integer. OPS/REXX always converts numeric values that fit in a signed fullword (integer values in the range -2147483648 through 2147483647) to binary values and performs register arithmetic on them irrespective of the NUMERIC DIGITS setting. For NUMERIC DIGITS settings, up to and including 10, in cases where the numeric value will fit in a signed fullword, OPS/REXX internally uses binary register arithmetic. This is one of the reasons why OPS/REXX significantly outperforms IBM REXX.

  The following example illustrates this difference:

  ```
  /* REXX */
  numeric digits 5
  a = 2147483646+1
  say a
  ```

  OPS/REXX output is: 2147483647

  IBM REXX output is: 2.1475E+9

  Note: OPS/REXX is not strictly correct in its numeric processing but the performance advantages are significant and CA does not intend to change OPS/REXX to comply with the REXX language definition in this case. For performance reasons CA recommends that, unless your REXX program requires greater precision, you allow NUMERIC DIGITS to default to 9.

Similarities Between OPS/REXX and Standard REXX

Both OPS/REXX and the standard REXX language enable you to issue commands to various host environments. Both versions of REXX offer symbolic substitution that is simpler than in the TSO/E CLIST language or in z/OS JCL.

The current version of OPS/REXX supports these standard REXX features:

- All REXX programming structures as defined in the book *THE REXX LANGUAGE: A Practical Approach to Programming* by M.F. Cowlishaw. For example, OPS/REXX supports counter variables on DO statements and the PROCEDURE statement.
- All standard SAA REXX functions plus most of the functions documented in the second edition of the Cowlishaw book, except for the I/O functions (CHARIN, CHAROUT, CHARS, LINEIN, LINEOUT, and LINES).
- Numbers with decimal points and exponents, as well as numeric digits with a precision up to 20 (default 9).
- Strings containing as many as 32,000 characters, including strings to ISPEXEC to support long commands and values of all REXX variables including global variables. OPS/REXX can build dynamic display areas in panels.

  The CA OPS/MVS REXXMAXSTRINGLENGTH parameter enables you to use a lower maximum string length if you want.

Differences Between OPS/REXX and Standard REXX

In addition to the similarities listed above, there are several important differences between OPS/REXX and standard REXX. These include:

- In OPS/REXX, external subroutines are resolved and bound with the main program prior to execution. This characteristic provides a major performance benefit for OPS/REXX when calling external subroutines, particularly in the AOF rule environment. One negative aspect of this characteristic is that all subroutines must be available at the time an OPS/REXX program or AOF rule is compiled or enabled on any system, even if some subroutines are never actually called during execution in that environment. For example, consider the following code:

  ```
  if OPSINFO("SMFID") == "SYSA" then
    call EXTSUB1
  else
    call EXTSUB2
  ```

  Clearly, the EXTSUB1 subroutine is called only when the code executes on SYSA. Nevertheless, OPS/REXX requires the EXTSUB1 subroutine (or load module) to be available on every system. In TSO/E REXX, external subroutines are resolved only when they are called during execution.

  In CA OPS/MVS, you can use the OPSWXTRN keyword of the OPTIONS instruction to indicate to OPS/REXX which external subroutines, built-in functions, and load modules are not absolutely required to be present prior to execution. The presence of the OPSWXTRN keyword in an OPTIONS instruction allows programs containing this OPTIONS instruction to be used by both OPS/REXX and TSO/E REXX, so the portability of REXX code that uses this instruction is unaffected. For more information, see OPTIONS Instruction in this chapter.

- In the Cowlishaw definition of REXX, when a PULL instruction is executed and the external data queue is empty, a read is done from the default character input stream. In OPS/REXX, this is not practical because in a rule, the only possible default character input stream is the console. Prompting the operator for the next line of input would be undesirable to say the least. Therefore, in OPS/REXX a PULL on an empty external data queue results in a null (zero length) line being returned.

- The PUSH instruction is not implemented in OPS/REXX. Its use results in REXX error number 64, which is the unimplemented feature error. The QUEUE instruction is implemented in OPS/REXX, and in most cases, you can use it to accomplish the same results.

- Because OPS/REXX is required to run efficiently in different environments, some differences exist in REXX instructions such as the INTERPRET instruction, where a limited implementation is available. Since the INTERPRET instruction causes the compiler to be reinvoked, any storage required to process the instruction is retrieved from existing sources, to avoid dynamically getting storage. In most cases the VALUE function would serve as a much more efficient alternative to INTERPRET.

- When converting from date format B to any other format, standard REXX adds one day to the resulting date, whereas OPS/REXX does not.

Note: For more information see the OPS/REXX DATE function in the *Command and Function Reference*.

Characteristics of OPS/REXX Programs

You can execute an OPS/REXX program in any of these ways:

- Explicitly, through the OPSEXEC (or OX) command
- Implicitly, through the OPSIMEX (or OI) command
- As batch jobs
- As started tasks

Use the OPSEXEC and OPSIMEX commands to execute OPS/REXX programs in source code or precompiled format. Use the OXCOMP and OICOMP commands to compile OPS/REXX programs without executing them.

When used with the AOF, OPS/REXX programs have a special structure and are called rules. Outside the AOF environment, OPS/REXX programs are called programs.

Unless you have precompiled rules, the CA OPS/MVS product compiles rules when you activate them with the ENABLE command and runs them strictly from their internal form (rather than reloading and reinterpreting them each time they are needed). Outside the AOF environment, OPS/REXX programs execute from source code or from staged internal forms.

Differences Between Precompiled and Source REXX Programs

The CA OPS/MVS product has always supported running OPS/REXX programs in source code format. Now, the product also supports storing and running precompiled OPS/REXX programs. You can issue OXCOMP and OICOMP commands to compile OPS/REXX programs without executing them; both commands store the compiled REXX code in the data set allocated to the OPSCOMP ddname. You can copy data sets containing precompiled OPS/REXX programs with no restrictions.

Precompiled OPS/REXX programs use exactly the same REXX functions as source code programs, but the precompiled programs are loaded faster. When you call a precompiled program, OPS/REXX loads and executes it, eliminating the time needed to read the source code and compile it. In contrast, source code programs take more time to load because OPS/REXX has to read and compile the code first.

Explicit and Implicit Program Execution

The only difference between implicit and explicit program execution is how you specify the name of the program to execute and where OPS/REXX looks for the name:

- With explicit execution, you supply the name of the data set containing the program.
- With implicit execution, you provide only a program name. OPS/REXX then locates that program in the library or libraries allocated to ddname OPSEXEC or SYSEXEC.

Formats for OPS/REXX Data Sets

Data sets containing OPS/REXX programs can have RECFM=V or RECFM=F record formats, with or without line numbers. For variable records, OPS/REXX checks columns 1 through 8 of the first line in the program to determine if line numbers exist. For fixed records, OPS/REXX checks the last eight columns of the first record to determine if line numbers exist.

Note: This checking follows standard TSO conventions.

If the first line in the program has no line number, be sure to check that no subsequent lines have numbers. This can occur if someone uses ISPF EDIT to copy lines from a line-numbered program into a program without line numbers.

The maximum logical record length supported is LRECL=255.

How OPS/REXX Locates Stored OPS/REXX Programs

OPS/REXX programs can reside in OPSEXEC data set libraries or in data sets elsewhere in the SYSEXEC concatenation. Programs in data sets allocated to OPSEXEC execute faster because OPS/REXX can execute them without compiling them first.

When you invoke a program, OPS/REXX always looks for it first in the OPSEXEC libraries. This is true as long as you have allocated the OPSEXEC ddname. If the program is not there, OPS/REXX looks in the SYSEXEC libraries and, after finding it, reads, compiles, and runs it.

Note: To avoid slowing down CA OPS/MVS processing, use the OPSEXEC DD, which prevents REXX code from being compiled at runtime.

Execute a Program That Calls External Routines

Compiling an OPS/REXX program also compiles all subroutines associated with it.

The way CA OPS/MVS scans for external routines depend upon whether you are editing a sequential data set or the member of a partitioned data set.

When you explicitly execute an OPS/REXX program that is not precompiled and that resides in a partitioned data set, CA OPS/MVS scans that data set to try to locate any external routines. If external routines are not found in the data set containing this program, CA OPS/MVS then searches in the libraries concatenated under the SYSEXEC ddname.

When you explicitly execute a program residing in a sequential data set, CA OPS/MVS scans only the SYSEXEC ddname for external routines.

To explicitly execute a program, use either the !OI or !OX edit macro.

Use the Precompiled OPS/REXX Programs

The following section discusses the use of precompiled OPS/REXX programs.

The OPSEXEC and OPSCOMP Libraries

As described earlier, when you invoke a program, OPS/REXX first looks for a precompiled version of it in the OPSEXEC libraries. If the program is found, OPS/REXX loads and executes it. If OPS/REXX cannot find the program in the OPSEXEC libraries, it looks in the SYSEXEC libraries and, after finding it, reads, compiles, and runs it.

If you plan to precompile and execute OPS/REXX programs, you must allocate two different ddnames, OPSCOMP and OPSEXEC. If you never intend to precompile OPS/REXX programs, but you do plan to execute them, you need to allocate only the OPSEXEC ddname.

Allocate the OPSCOMP DDname

The OPSCOMP ddname contains the output of the compile process. When you compile an OPS/REXX program using either the OICOMP or OXCOMP command, CA OPS/MVS saves the compiled version of the program as a member in the data set you have allocated to the OPSCOMP ddname. Because the OPSCOMP ddname is an output data set, it must be a single data set rather than a concatenation. This rule conforms to standard z/OS restrictions on output data sets.

To allocate the OPSCOMP data set, use either the JCL or the TSO ALLOC command shown here:

- JCL

  ```
  //OPSCOMP DD DISP=SHR,DSN=user.compile.library
  ```

- TSO

  ```
  ALLOC FILE (OPSCOMP) DATASET('user.compile.library') SHR
  ```

In the previous examples, *user.compile.library* is the name of the data set that is to store the output of the compile process. Different *user.compile.library* data sets can be in use at different times.

Follow these guidelines when allocating OPSCOMP:

- For additional flexibility, do not allocate OPSCOMP permanently.
- Allocate OPSCOMP to an output partitioned data set with a minimum block size of 4096 (at least 8192 is recommended) and a logical record length of 4096. You may want to reblock the data set for efficiency; the larger the block size, the better the performance. The OPEXRBLK member of the OPS.CCLXCNTL library contains JCL to reblock this data set or any compiled library.
- Place the data set you allocate to the OPSCOMP DDNAME *first* in your OPSEXEC concatenation (as described below).

Allocate the OPSEXEC DDname

When you invoke an OPS/REXX program, OPS/REXX first searches the OPSEXEC ddname for a precompiled version of it.

For this reason, the same data set that you allocated to the OPSCOMP ddname typically should also be the first data set in your OPSEXEC ddname concatenation.

If you are an OPSVIEW user, second in your OPSEXEC ddname concatenation should be the SYS1.OPS.CCLXOPEX data set. The SYS1.OPS.CCLXOPEX data set stores all previously compiled REXX programs for the OPSVIEW feature, and should be allocated to the OPSEXEC ddname of all OPSVIEW users for improved performance. In case you have customized any of these programs for use at your site, placing this CA-distributed data set second ensures that OPS/REXX finds your versions first.

The SYS1.OPS.CCLXOPEX data set has a block size of 8192. For performance reasons, you may want to re-block the SYS1.OPS.CCLXOPEX data set (the larger the block size, the better the performance). To do so, use the JCL in the OPEXRBLK member of the OPS.CCLXCNTL library.

To allocate the OPSEXEC ddname, use either the JCL or the TSO ALLOC command shown in the following examples.

Note: In these examples, the SYS1.OPS.CCLXOPEX data set appears second in the concatenation:

- JCL

```
//OPSEXEC DD DISP=SHR,DSN=user.compile.library
//        DD DISP=SHR,DSN=SYS1.OPS.CCLXOPEX
```

- TSO

```
ALLOC FILE (OPSEXEC)
  DATASET('user.compile.library' 'SYS1.OPS.CCLXOPEX') SHR
```

In the examples, *user.compile.library* is the name of the data set where you store your OPS/REXX programs. Different *user.compile.library* data sets can be in use at different times.

Maintain Compiled OPS/REXX Programs

At times, internal changes in OPS/REXX can make all of your precompiled REXX programs and rules incompatible with the current OPS/REXX execution environment. When this occurs, CA OPS/MVS issues message OPS0990 to warn you to recompile your REXX programs and rules.

To maintain compiled OPS/REXX programs

1. Use OPSVIEW options 2.4 and 2.5 to do maintenance on precompiled REXX programs.
2. Use OPSVIEW options 2.5 and 4.5.2 to help you recompile.

OICOMP Command

Using the CA OPS/MVS OICOMP command, you can compile any program in the SYSEXEC ddname and save it in the data set allocated to the OPSCOMP ddname. When using OICOMP, you need to provide only a program name.

Use either of these OICOMP formats:

- Format 1

  ```
  OICOMP member [NOSOURCE]
  ```

- Format 2

  ```
  OICOMP PROGRAM(member) [NOSOURCE]
  ```

OXCOMP Command

The OXCOMP command enables you to compile any program in any partitioned or sequential data set and save it in the data set allocated to the OPSCOMP ddname. When using OXCOMP, you need to provide the name of the data set containing the program. You can also use OXCOMP to compile an entire partitioned data set by omitting the specification of a value for *member*. If you are compiling a particular program in a partitioned data set, be sure to include the name of the program as the value for *member*.

Use either of these OXCOMP formats:

- Format 1

  ```
  OXCOMP 'dsname(member)' [NOSOURCE]
  ```

- Format 2

  ```
  OXCOMP PROGRAM('dsname(member)') [NOSOURCE]
  ```

Values You Specify for OICOMP and OXCOMP

The *member* value specifies the name of the member containing the OPS/REXX source program. The NOSOURCE keyword causes the compiled version of the program to be saved without any SOURCE statements, generating a smaller data set member that can be loaded into storage more quickly. If you specify NOSOURCE, TRACE statements and error messages from the SYNTAX routine do not include SOURCE statements, and the REXX SOURCELINE() function is unusable. Instead, OPS/REXX displays the number of the line containing the statement in error.

Execute Source OPS/REXX Programs

The following section discusses the execution of source OPS/REXX programs.

Ways to Invoke OPS/REXX Programs in Source Format

You can invoke OPS/REXX programs in source code format as follows:

- From READY mode in an interactive TSO session
- From ISPF dialogs
- From ISPF EDIT
- From AOF rules
- From other OPS/REXX programs
- In a batch job, by directly executing the OPS/REXX interpreter
- In a batch job, under the batch TSO TMP
- From programs written in other languages

The following sections describe each of these OPS/REXX program invocations in detail.

Explicitly Compared to Implicitly Specifying the OPS/REXX Program Data Set

Using the OPSEXEC (alias OX) command executes an OPS/REXX source program *explicitly*, while using the OPSIMEX (alias OI) command invokes a source program *implicitly*.

Issue the OPSEXEC (OX) Command

The OPSEXEC or OX command has the following syntax:

```
OPSEXEC|OX PROGRAM('dsname[member]')
  [ARG('arguments')]
  [ITRACE('x')]
  [MAXEDQ(lines)]
  [SUBSYS(ssid)]
  [WORKSPACE(size)]
```

To execute an OPS/REXX source program directly from TSO READY mode (or ISPF PDF menu 6) using the OPS/REXX interpreter, use either of the following formats for OPSEXEC:

```
OPSEXEC "dsname" argument
OX "dsname" argument
```

For detailed explanations on each OPSEXEC keyword, see OPSEXEC Command Processor in the *Command and Function Reference*.

Issue the OPSIMEX (OI) Command

You can execute any source program located in ddname SYSEXEC directly from TSO READY mode (or ISPF PDF Menu 6) using the OPSIMEX (alias OI) command. This command executes programs implicitly. You must preallocate the SYSEXEC file name to one or more partitioned data sets before issuing the command.

The OPSIMEX or OI command has the following syntax:

```
OPSIMEX|OI PROGRAM('dsname[member]')
  [ARG('arguments')]
  [ITRACE('x')]
  [MAXEDQ(lines)]
  [SUBSYS(ssid)]
  [WORKSPACE(size)]
```

You can also use these abbreviated formats for OPSIMEX:

```
OPSIMEX member argument
OI member argument
```

The OPSIMEX command uses exactly the same keywords and arguments as the OPSEXEC command; the only difference is that OPSIMEX invokes a program implicitly. When you invoke OPS/REXX implicitly, the argument string is passed exactly as it is passed when you invoke OPS/REXX explicitly.

Note: When using the form of the OPSIMEX or OI command that uses PROGRAM or other keywords, enclose the argument string in single quotation marks.

The record formats supported for the SYSEXEC libraries are the same as those supported for REXX data sets used to invoke OPS/REXX explicitly.

Implicitly Invoke Source Programs

To invoke a REXX program called ABC with a null argument, issue the following command:

```
OI ABC
```

To invoke the same program with an argument of *Now is the time*, issue either of the following commands. These commands work only if the member ABC exists in the library or libraries allocated to either ddname OPSEXEC or ddname SYSEXEC.

```
OI ABC Now is the time
OI PROGRAM(ABC) ARG("Now is the time")
```

Execute OPS/REXX Source Programs from ISPF Dialogs

You can invoke OPS/REXX source programs from in ISPF dialogs explicitly or implicitly as shown in the following example. Implicitly invoking a program, however, uses less overhead.

```
SELECT PGM(OI) PARM(member argument)

SELECT CMD(OI PROGRAM (member) [ARG(argument)] [SUBSYS(ssid)])

SELECT PGM(OX) PARM(member argument)

SELECT CMD(OX PROGRAM (dataset(member)) [ARG(argument)] [SUBSYS(ssid)])
```

Note: If you use PGM calls to issue the OX or OI command, you can specify the subsystem ID as an argument.

The PGM calls differ from the CMD calls only in the way parameters are passed. However, for CMD calls, ISPF Dialog Management services move the cursor to the bottom of the screen before the command starts to execute. This can be annoying; to avoid it, use PGM calls.

The OPS.CCLXEXEC library distributed with CA OPS/MVS contains examples of writing complex ISPF dialogs in OPS/REXX.

Execute OPS/REXX Source Programs from ISPF EDIT

If you are editing an OPS/REXX program using ISPF EDIT, you can execute it directly under ISPF EDIT using one of the following forms of the OX ISPFEDIT macro call. You can omit the exclamation point that precedes the OX or OI command if the following statement appears in your initial ISPF EDIT macro:

```
ISREDIT DEFINE OX PGM MACRO
```

```
!OX
!OI
!OX "argument"
!OI "argument"
```

Entering any of these commands on the command line while in ISPFEDIT causes the current ISPFEDIT data set to execute. The program is not saved to disk; instead, the OPS/REXX interpreter uses the program copy being kept by ISPFEDIT in virtual storage. This feature is especially handy when you are debugging OPS/REXX programs.

For details about ISPF EDIT macros, see *ISPF/PDF Edit and Edit Macros*.

How OPS/REXX Programs in the AOF Work

In the AOF environment, the OPS/REXX interpreter does the following:

- Compiles an AOF rule when it is enabled if no precompiled version of that rule exists. While compiling a rule, the interpreter reads the rule source code from the rule data set library on disk and places the code in internal CA OPS/MVS storage. The interpreter does the same for any external subroutines that are called directly or indirectly.
- Runs the)INIT section of a rule when a rule is being enabled.
- Runs the)TERM section of a rule when a rule is being disabled.
- Runs the)PROC section of a rule when an event matching the event type specified for the rule occurs.

All other uses of OPS/REXX require you to invoke the OPS/REXX interpreter explicitly.

Call the Interpreter from an OPS/REXX Program

An OPS/REXX program can call the OPS/REXX interpreter as an OPS/REXX function or subroutine. This calling method is not efficient, but it can be useful in special situations.

Use either of the following statements to call the interpreter from another OPS/REXX program:

```
variable = OI("member argument")
CALL OI "member argument"
ADDRESS TSO
  "OI member argument"
  "OX dsname argument"
```

The value returned by the function calls (or into variable RESULT by the CALL instructions) is the error code of the interpreter. Usually, this value is the same as any standard REXX error code (or zero if the program was interpreted without errors).

When you use the ADDRESS TSO form, any messages and output produced by the program (including TRACE output) are returned in the external data queue just like other TSO commands (unless the calling program is executing under an AOF rule).

Execute OPS/REXX Programs from Batch

You can invoke OPS/REXX programs implicitly or explicitly from batch.

Invoke an OPS/REXX Program Implicitly

To invoke an OPS/REXX program implicitly and run it as a batch job, use the following JCL:

```
//stepname EXEC PGM=OI,PARM='member argument'
//OPSEXEC DD DSN=user.compiled.library
//        DD DSN=OPS.SYS.CCLXOPEX
//SYSEXEC DD DSN=library,DISP=SHR
```

The DSN parameter must give a fully qualified data set name without quotation marks. No prefix or suffix is added to the name you specify.

Invoke an OPS/REXX Program Explicitly

To invoke an OPS/REXX program explicitly and run it as a batch job, you need only the JCL shown here:

```
//stepname EXEC PGM=OX,PARM='dataset(member) argument'
```

Considerations for Batch Execution

Before executing an OPS/REXX program as a batch job, note that:

- Positive numeric values specified on the RETURN statement are retrieved as the return code.
- You cannot use the ADDRESS TSO and ADDRESS ISPEXEC environments in batch mode.
- Batch programs may be a better place to run long-running REXX programs than servers, which do best at processing short-term jobs.
- When an OPS/REXX program runs as a batch job or as a started task and you are either running multiple CA OPS/MVS copies on one system or the subsystem where CA OPS/MVS runs is not called OPSS, you can ensure that the program executes on the correct subsystem. To do so, specify the following when invoking the program or submitting the batch job:

  ```
  OPS$ssid DD DUMMY
  ```

  The *ssid* is the subsystem name.

Execute OPS/REXX Programs from Batch (Under the Batch TSO TMP)

Use the following JCL to run the TSO TMP as a batch job step:

```
//          EXEC PGM=IKJEFT01
//          PARM='first command or CLIST to be executed'
//SYSPROC   DD DISP=SHR,DSN=user.clistlib
//SYSEXEC   DD DISP=SHR,DSN=user.rexxlib
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD *
   (TSO commands or clists to be executed
                  * including *
    OX commands to invoke OPS/REXX programs from data sets,
    OI commands to invoke OPS/REXX programs from //SYSEXEC,
    or both)
/*
```

Execute OPS/REXX Programs from USS

OPS/REXX programs can be executed in the USS shell by invoking the OI or OX command from an HFS or zFS in compatibility mode file. The only restriction is that TSO must not be active in the same address space in which OPS/REXX is invoked. For an OMVS session invoked from TSO or ISPF, the parameter NOSHAREAS must be specified with the OMVS command. This ensures that the USS shell actually runs in a USS transaction server address space. Command syntax parsing in USS is also different than in TSO. USS parses the parameter string that follows the command file name for imbedded variable substitution and other USS commands. Therefore the parameters destined for OPS/REXX must be enclosed in single or double quotes. Double quotes are not as restrictive as single quotes and may result in the need to use the escape character to prevent undesired substitutions.

While OX dynamically allocates the data set containing the desired OPS/REXX program, OI assumes a pre-allocated SYSEXEC data set. SYSEXEC must be allocated in the USS transaction server address space for OI to work. Although OPSDYNAM is an OPS/REXX host command, it can also be used as a USS shell command to allocate the required SYSEXEC data sets. Other POI commands that can execute as USS shell commands are OPSGETV, OPSGETVL, OPSSETV, OPSDELV, and OPSQL. All other OPS/REXX functions and host commands such as ADDRESS WTO are available in an OPS/REXX program run as a USS command using OI or OX.

There are two ways to place CA OPS/MVS commands in the USS HFS or zFS in compatibility mode file system. The first method is to copy the desired commands with all their aliases to a PDSE data set, and then use the TSO OPUT command to copy the PDSE load modules to the desired HFS or zFS in compatibility mode directory. In the case of OPS/REXX, you would use ISPF 3.3 to copy module OPSIMEX and all its aliases (OI, OICOMP, OIDB, OPSEXEC, OX, OXCOMP, OXDB, and OXSCAN) to a PDSE. Then issue the OPUT command from TSO for any aliases you want to place in the HFS or zFS in compatibility mode:

```
OPUT 'OPS.PDSELOAD(OX)'  '/opsmvs/ox'  BINARY
OPUT 'OPS.PDSELOAD(OI)'  '/opsmvs/oi'  BINARY
```

The disadvantage of this method is that maintenance applied to the product must be propagated manually to the HFS or zFS in compatibility mode.

The second method is to create null HFS or zFS in compatibility mode files for the commands and use the sticky bit feature of HFS or zFS in compatibility mode files to cause USS to search for the real load modules from STEPLIB and LINKLIST/LPA. To create the null files, issue the following USS command:

```
touch  /opsmvs/oi
```

To set the sticky bit for the file, issue the USS command:

```
chmod o=+s  /opsmvs/oi
```

To verify the permissions of the command files, issue the USS command:

```
ls -EH /opsmvs/oi
```

The advantage of this method is that no special procedures are required to recopy module maintenance, no PDSE is required for copies, and no significant HFS or zFS in compatibility mode file space is used. However, when the sticky bit method is used, dynamic allocations using OPSDYNAM as a command to allocate SYSEXEC is not retained for subsequent commands. In this case the OI command is of no practical use and only OX can be used.

To execute the OPS/REXX commands in the HFS or zFS in compatibility mode from a Telnet or OMVS NOSHAREAS session, the CA OPS/MVS load library must be in the system LINKLIST/LPA or a STEPLIB must be used. For example, you could use the following sequence of USS shell commands to execute the MYEXEC OPS/REXX program:

```
export STEPLIB=SYS1.OPS.CCLXLOAD
/opsmvs/ox  "'SYS2.OPS.REXX(MYEXEC)'"
```

Following is an example of how to call OPSQL directly:

```
/opsmvs/opsql "SELECT RULENAME FROM AOF_PERIODS WHERE NODE='XE33'"
```

To use OPS/REXX in the IBM USS REXX, OI and OPSDYNAM can be called as functions if they are located in the LINKLIST/LPA or STEPLIB. The argument list consists of a single argument using the same syntax as the TSO command. The following USS REXX program invokes OI to execute an OPS/REXX program:

```
/*    Rexx    */
/*  Invoke OPS/REXX program  TESTINFO */
opsrc = OPSDYNAM("ALLOC DD(SYSEXEC) DSN(SYS2.O.REXX) SHR REUSE")
 say "Calling OPS/REXX"
opsrc = OI("TESTINFO")
do while QUEUED() > 0
  pull qmsg
  say "XDQ="qmsg
end
return opsrc
```

The following USS REXX program invokes OPSQL as a function:

```
/*    Rexx    */
z = OPSQL("SELECT RULENAME FROM AOF_PERIODS WHERE NODE='XE33'")
 say 'RC'z
if RULENAME.0>0 then
do I=1 to RULENAME.0
  say RULENAME.i
end
```

The Interaction of OPS/REXX with Other Languages

One of the most powerful features of OPS/REXX is its extendability by users. You can write OPS/REXX external subroutines and functions in assembler language; source module OPSMFID in the ASM library shipped with all copies of CA OPS/MVS provides an excellent reference for users wanting to do so.

Requirements for Non-REXX External Functions

An OPS/REXX external function or routine not written in REXX must meet these requirements:

- It must be a load module. The CA OPS/MVS product uses the z/OS LOAD macro to load the external routine.
- If the routine is called from an AOF rule:
  - The load module must be reentrant.
  - The code must be able to run in cross-memory mode.
  - The code cannot issue SVC calls (such as GETMAIN, WTO, I/O, and so on).
  - The code cannot acquire resources from the home address space.

These restrictions are inflexible and require in-depth knowledge of cross-memory mode programming. In many cases, you must move data to areas accessible in cross-memory mode, suspend cross-memory mode to be able to call a system service, or both.

Register Contents

On entry to a REXX function, the registers contain the following:

- Register 0 contains the address of an ENVBLOCK
- Register 1 contains the address of an EFPL
- Registers 2 through 12 are unpredictable
- Register 13 points to a save area for the routine
- Register 14 contains the return address to the caller
- Register 15 contains the entry point address of the routine

EFPL Format

The External Function Parameter List (EFPL) format is compatible with the IBM TSO/E REXX EFPL and can be mapped by the TSO/E REXX macro IRXEFPL. OPS/REXX supports only two fields: EFPLARG, which points to the argument, and EFPLEVAL, which points to the word containing the address of the result EVALBLOCK (OPS/REXX supports all the fields in the EVALBLOCK). The TSO/E REXX macro IRXEVALB maps the EVALBLOCK.

The save area pointed to by register 13 is 72 bytes long. To store the contents of the registers on entry in this save area, use the z/OS SAVE macro.

Outcome of Processing a REXX Routine

When the REXX routine has completed processing, it should do the following:

1. Store a function result in the EVALBLOCK pointed to by the EFPL. This is required for a function call and optional for routines called through the REXX CALL instruction.
2. Restore all registers to the state they were in upon entry to the routine.
3. Set a return code in register 15.
4. Return control to the address passed in register 14 on entry.

Pass Arguments

Field EFPLARG in the EFPL (whose address is passed in register 1 on entry to a REXX function) contains the address of the argument list. The argument list consists of two full words (eight bytes) for each argument specified on the routine or function call. The first word points to the address of a string and the second word contains the length of the string. X'FFFFFFFFFFFFFFFF' terminates the argument list.

For example, this function call:

```
a = UserFunc("First",2,"Argument 3")
```

Results in this argument list:

```
DC A(ARG1),F"5"
DC A(ARG2),F"1"
DC A(ARG3),F"10"
DC X"FFFFFFFF",X"FFFFFFFF"
```

Pointed to by field EFPLARG. In this example:

```
ARG1 DC C"First"
ARG2 DC C"2"
ARG3 DC C"Argument 3"
```

Omit Arguments

A value of zero in the first word of the argument and a zero in the second word denotes an omitted argument. An omitted argument is different from a null string argument (indicated by a non-zero first word and a zero in the second word). For example, this function call:

```
x = abc(,"",3)
```

Results in this argument list:

```
DC A(0),F"0"
DC A(ARG2),F"0"
DC A(ARG3),F"1"
DC X"FFFFFFFF",X"FFFFFFFF"
```

In this example:

```
ARG2 DC C" " or  ARG2 EQU 1
ARG3 DC C"3"
```

The "" symbol denotes a null argument.

Return Information

On entry to an OPS/REXX program, the field EFPLEVAL in the external function parameter list (the EFPL whose address is passed in register 1) points to an EVALBLOCK, an evaluation control block. The evaluation control block is an area the caller preallocated to return a function result. A routine need not return a result if a REXX CALL instruction invoked it; but a function result must be returned in all other cases.

Copy the result (a string) into the EVALBLOCK_EVDATA. Check the maximum length of this field to make sure that you do not overlay storage. The EVALBLOCK that CA OPS/MVS builds can contain a 32,000-byte result value.

The length of the returned string should be stored as a fullword integer in field EVALBLOCK_EVLEN. On entry to a routine, the caller has primed this field with the value X'80'. If this value returns unchanged, the function returns no result.

Send Data to the External Queue

User-designed functions written in assembler language can send data to the OPS/REXX external data queue using the standard TSO/E REXX IRXSTK service. You can get the address of the REXX stack service routine from the EXTE pointed to by the environment block (register 0 on entry to the function). QUEUE is the only supported function.

For more information on the stack service, see the IBM documentation.

Create REXX Variables

User-written functions written in assembler language can create and extract REXX variables using the standard TSO/E REXX IRXEXCOM service. You can get the address of the REXX variable service routine from the EXTE pointed to by the environment block (register 0 on entry to the function). QUEUE is the only supported function.

For more information on the IRXEXCOM service, see the IBM documentation.

OPS/REXX Execution Limits

The following section discusses OPS/REXX execution limits.

Resource Use Monitoring

When OPS/REXX executes a REXX program, it checks to see that a program does not consume an excessive amount of resources. This checking is especially important for executing AOF rules that process system events, as runaway rules can degrade system performance significantly, causing poor response time and other problems.

Specifically, OPS/REXX monitors:

- Program execution time
- How many REXX clauses executed
- How many REXX SAY instructions executed
- How many host commands were issued
- How many output lines the external data queue contains

Parameters That Set Limits

CA OPS/MVS product parameters set limits for the execution values listed on the previous page. You can limit these values separately for AOF rules and for other REXX programs run through OX and OI commands.

The following parameters set limits:

AOFMAXSECONDS

For AOF rules except request rules, sets the maximum time, in seconds, that a rule can execute for a given event.

AOFMAXCLAUSES

For AOF rules except request rules, sets the maximum number of clauses that a rule can execute for a given event.

AOFMAXSAYS

For AOF rules except request rules, sets the maximum number of SAY instructions that a rule can execute for a given event.

AOFMAXCOMMANDS

For AOF rules except request rules, sets the maximum number of host commands that a rule can execute for a given event.

AOFMAXQUEUE

For AOF rules including request rules, sets the maximum number of lines that a rule can have in the external data queue for a given event.

REXXMAXSECONDS

For request rules and REXX programs, sets the maximum time, in seconds, that a REXX program or request rule can execute for a given event.

REXXMAXCLAUSES

For request rules and REXX programs, sets the maximum number of clauses that a REXX program or request rule can execute for a given event.

REXXMAXSAYS

For request rules and REXX programs, sets the maximum number of SAY instructions that a REXX program or request rule can execute for a given event.

REXXMAXCOMMANDS

For request rules and REXX programs, sets the maximum number of host commands that a REXX program or request rule can execute for a given event.

REXXMAXQUEUE

For REXX programs, sets the maximum number of lines that a REXX program can have in the external data queue for a given event.

REXXMAXSTRINGLENGTH

For request rules and REXX programs, sets the maximum length of any string in a REXX program or request rule.

Important! OSF TSO servers are not intended for running an OPS/REXX program that takes a long time to execute. You should use OSF TSL servers for those programs. When an OPS/REXX program running on an OSF TSO server exceeds the server execution limits set by the CA OPS/MVS OSFCPU, OSFOUTLIM, OSFRUN, or OSFWAIT parameters, the OSF terminates that program even if it has not exceeded any of the AOF or REXX execution limits described above. For example, both the OSFRUN parameter and the REXXMAXSECONDS parameter specify how long a rule or REXX program can take to execute. So, if the value of OSFRUN is lower than the REXXMAXSECONDS value, the OSF stops a program executing on an OSF TSO server as soon as it exceeds the time limit that OSFRUN set.

To prevent programs executing in OSF TSO servers from terminating prematurely, either raise the values of the CA OPS/MVS OSFCPU, OSFOUTLIM, OSFRUN, and OSFWAIT parameters or run the program as a separate started task or batch job. Equivalent parameters (OSFTSLCPU, OSFTSLOUTLIM, OSFTSLRUN, and OSFTSLWAIT) control the limits for the OSF TSL servers and (OSFTSPCPU, OSFTSPOUTLIM, OSFTSPRUN, and OSFTSPWAIT) control the limits for the OSF TSP servers.

Override Execution Limits

OPS/REXX programs and AOF rules can override most of the execution limits by issuing the REXX OPTIONS instruction. However, you cannot change the maximum number of lines allowed in the external data queue once a REXX program begins executing. For a more detailed discussion, see OPTIONS Instruction in this chapter.

Elements of OPS/REXX

The following section discusses the elements of OPS/REXX.

Symbolic Substitution in OPS/REXX

If you are familiar with z/OS JCL and the TSO CLIST language, you will notice that OPS/REXX does not use ampersands (the & symbol). Symbolic substitution in REXX is different from that in the TSO CLIST language or in z/OS JCL, and is much easier to use.

REXX Elements That OPS/REXX Supports

OPS/REXX implements all of the elements of the SAA standard REXX language except for the following instructions and functions:

- FORM
- OPTIONS ETMODE
- PUSH
- The input/output functions CHARIN, CHAROUT, CHARS, LINEIN, LINEOUT, LINES, and STREAM
- SCAN portion of TRACE

In most cases, using an unsupported feature results in OPS/REXX error code 64 - UNIMPLEMENTED FEATURE.

Implementation Limits

The following table describes the minimum limits that SAA imposes and the maximum limits that OPS/REXX imposes on REXX elements:

Item	SAA	OPS/REXX
Literal strings	100 bytes	32000 bytes
Symbol (variable name) length	250 bytes	250 characters for local symbols; 50 characters for global symbols; 32 characters for function and routine name labels
Nesting control structures	100	Varies, depending upon the complexities of the structures that are involved
Call arguments	20	10
MIN and MAX function arguments	20	20
Queue entries	100	Controlled by parameter; defaults to 3000
NUMERIC DIGITS value	1000	No maximum limit
Notational exponent value	999 999 999	999 999 999
Hexadecimal strings	250 bytes	32000 bytes
C2D input string	Limit is either 250 or the NUMERIC DIGITS value divided by 2, whichever is smaller	4 (must be a positive number)

Item	SAA	OPS/REXX
D2C output string	Limit is either 250 or the NUMERIC DIGITS value divided by 2, whichever is smaller	256
X2D output string	Limit is either 500 or the NUMERIC DIGITS value minus 1	32000

Constants in OPS/REXX

OPS/REXX supports character strings up to 32,000 characters long.

Numeric values can include decimal points and exponential notation.

Symbols in OPS/REXX

Local symbols can contain up to 250 characters; global symbols, up to 50 characters, and labels used in function or subroutine calls, up to 32 characters.

Variable Values

Variables containing character strings can contain no more than 32000 bytes (or a lower value set by the REXXMAXSTRINGLENGTH parameter). This limitation also applies to intermediate results.

Compound Symbols

The limits on the symbol name (pre-substitution) and the derived name (post-substitution) of a compound symbol and on the value that a compound symbol may take are the same as the limits for the name and value of a regular variable.

You must treat the environmental variables of the CA OPS/MVS product as compound symbols. For example, if you have defined the variable SYSID in a rule and you refer to MSG.SYSID elsewhere, the SYSID part of MSG.SYSID is interpreted as the content of the previously defined variable of the same name.

Arithmetic Values and Operators

OPS/REXX supports floating point arithmetic. Although you can set NUMERIC DIGITS higher than 9, higher values can impair performance.

OPS/REXX normalizes all values based on the NUMERIC digits setting before it performs any other operations with the value. Unless NUMERIC DIGITS is set to a value larger than the largest value in the program, the results will not be the same as in other REXX implementations.

OPS/REXX Considerations

Before using OPS/REXX, consider the following points:

- OPS/REXX uses the following search order to locate external functions as it compiles a program:
  1. Built-in functions.
  2. The SYSEXEC concatenation or the ddname concatenation as specified in the REXXDDNAME product parameter.
  3. When using non-REXX external functions, a LOAD issued for a module with the name of the external reference (that is, the standard z/OS load module search mechanism is used).
- Avoid using OPS as the first characters of the names of REXX functions that you create because these names may conflict with built-in function names that may be added to the product in future releases.

How to Implement Common Coding Guidelines

The REXX programming language coupled with many of the CA OPS/MVS Host Environments and functions let you create effective and efficient automated applications. When creating your AOF rules and OPS/REXX programs, you should establish common coding guidelines (upper or lower case,comment blocks,number of spaces to indent,and so on).

To begin implementing common coding standards within CA OPS/MVS automated applications, review the following coding guidelines:

1. Design a template to be used as a beginning comment block within all programs and rules.

   Include informative data fields such as purpose, related programs rules, logic outline, and so on.

   The following is an example beginning comment block template:

```
/*********************************************************************/
/*   Name       - Rule_or_pgm_name                                   */
/*   Purpose    - Brief sentence or two to identify what this AOF    */
/*                rule or OPS/REXX program accomplishes.             */
/*   Related    - List any other related rules pgms here.Include     */
/*                ruleset name if an AOF rule is related,and complete */
/*                PDS name if an OPS/REXX pgm. Such as:              */
/*                          TOD.CICSSHUT                             */
/*                          SYS2.OPSMVS.USER.REXX(CICSSHUT)          */
/*   Globals    - List any GLOBAL,GLVTEMP,or GLVJOBID variables      */
/*                being used within program. Such as:                */
/*                GLVTEMP1.PRIMARY.MUF - Contains system of the      */
/*                                       primary CA Datacom/AD       */
/*                                       MUF system                  */
/*                GLVTEMP1.PRODFAIL.jobname - Contains failure info  */
/*                                            of the specific jobname */
/*                                                                   */
/*   History    - 25 OCT 2010 DAG  - Original implementation        */
/*                22 JAN 2010 DAG  - Added logic to .............    */
/*                                   ...............................  */
/*   Notes      - This section would list detailed information of    */
/*                of this rule or program. Begin with more details   */
/*                of why the automation is needed. List an overview  */
/*                of the logic in an 1-x format. Such as:            */
/*                                                                   */
/*                Outline of the logic flow :                        */
/*                1. ABENDLOG AOF rule processes the triggering      */
/*                   IEF450I event and inserts desired event data    */
/*                   into a RDF table created within rule.           */
/*                2. A dynamic TOD rule created within the ABENDLOG  */
/*                   AOF rule will trigger this OPS/REXX program.    */
/*                3. Logic in this program will simply read the data */
```

```
/*                  stored in the RDF table, and write to some      */
/*                  preallocated sequential data set.               */
/*                                  .                               */
/*                                  .                               */
/*********************************************************************/
```

2. Create uniform comment blocks to be used before the instructions and logic.

   Adhere to the following conventions:

   - Add a number line as last comment line to assist when indenting for wrap around lines or specific instructions such a DO...END.
   - Use mixed case in your comment descriptions.
   - Optionally, comment the block sections of a program, such as:
     - Main processing
     - Sub-routines

   Example comment block used before instructions or logic:

```
/*-------------------------------------------------------------------*/
/* Uniform comment block to be placed prior to specific logic or     */
/* instructions ......................                               */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----7*/

/*-------------------------------------------------------------------*/
/*--------------------- Internal sub-routines -----------------------*/
/*-------------------------------------------------------------------*/

/*-------------------------------------------------------------------*/
/* subroutine name:                                                  */
/*  -Brief description of work performed in this subroutine......    */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----7*/
```

3. Create meaningful names for the variables based on the data that they will be assigned.

   Simple variables will be used widely across all your rules and programs. The more descriptive the names, the easier it will be to maintain the code.

4. Use upper and lower case consistently to create variable names.

   Various supplied CA OPS/MVS samples use lower case for simple variable names. For example:

```
cicsregion   = msg.jobname
abendcode    = WORD(msg.text,4)
prod_systems = 'SYSA SYSB SYSC'
```

   Adhere to the following case conventions:

   - Choose to upper case CA OPS/MVS reserved stemmed variable names GLVJOBID, GLOBAL,or GLVTEMP.

     For example:

```
cics_regions = OPSVALUE('GLOBAL1.CICSREGIONS','O')

payroll_contacts = OPSVALUE('GLOBAL1.PAYROLL.CONTACTS','O')
```

- If part of the stemmed variable name is that of a defined simple variable, then use the same case for the variable as it was originally created.
- Create clean coding of the OPSVALUE() by assigning the variable name and value to be updated to a simple variable that will act as a staging variable.

  For example:

```
cicsregion = msg.jobname
varname    = 'GLVTEMP1.'cicsregion'.SHUTDOWN'
setvar     = OPSVALUE(varname,'U','REQUESTED')

smfid     = OPSINFO('SMFID')
ipldate   = OPSINFO('IPLDATE')
ipltime   = OPSINFO('IPLTIME')
iplvolser = OPSINFO('IPLVOLSER')
iplinfo   = ipldate ipltime iplvolser
ipl_varname = 'GLVTEMP1.IPLINFO.'smfid
setvar      = OPSVALUE(ipl_varname,'U',iplinfo)
```

5. Pick your case standard for coding OPS/REXX and TSO/E REXX functions.

   Authors of CA OPS/MVS samples uppercase all function names and optional or required arguments. If a variable name is used within a function argument, then use the same case that it was created in, for example:

```
inits  = OPSJES2('I','INIT','*','A')
isitactive = OPSTATUS('A','I',jobname)
device = WORD(SUBSTR(record,device_loc),1)
smfid = OPSINFO('SMFID')
```

6. Establish case guidelines for host environment commands, both CA OPS/MVS host environments and other host commands.

   Mixed case is mainly used within many of the CA OPS/MVS samples when coding the keywords of the host environments, with the exception of the the SQL host environment,where uppercase is used for table and column names.

   For example:

```
address WTO
"Msgid(OPSNOTIFY) Text('Payroll schedule is late')",
   "Desc(1) Route(1)"

address OPER
"Command(D GRS,ANALYSE,BLOCKER) Nooutput"

address SQL
"Update STCTBL set DESIRED_STATE = 'DOWN' where",
   "TYPE = 'TESTCICS'"
```

```
address OPER
"Command(DB1ADIS DB(DB2CA11) LIMIT(*) LOCKS)",
  "Cmdwait(30) Stopresp(DSN9022I) Interval(0) Stopend(NO)"
```

7. For OPS/REXX or REXX instruction, establish consistent case style, indenting, and commenting using the following guidelines:

   - Choose either upper case or lower case as follows:
     - if..then..else or If..Then..Else
     - do..end or Do..End
     - select..when..otherwise or Select..When...Otherwise
     - parse var msgtxt or Parse Var msgtxt
   - Be consistent in the number of spaces you indent to begin instruction.
     - To begin instruction, use the same number of spaces to indent your do...end instructions.
     - Use the same number of spaces to indent each instruction within the loop. (such as two spaces).
   - Line up the end instruction with the associating do instruction. This is especially helpful within nested the do...end loops.
   - Place a comment with the end instruction to identify what loop is ending.

     This coding aid assists in the maintenance of your programs.

   Note: The same rules apply when coding the select..when..otherwise instruction.

   Example:

```
select
  when commands = 'HELP'    then call help /*Ident when 2 spaces*/
  when commands = 'IPL'     then call ipl  /*Line up all when's */
  when commands = 'JACT'    then call jact
  when commands = 'JSPOOL'  then call jspool
  when commands = 'WTORS'   then call wtors
  when commands = ''        then call help
otherwise Nop                             /*Line up with 'select'*/
end                                       /*Line up with 'select'*/

do i = 1 to total_msfnames          /* Loop through all abends  */
  pull msfedq                       /* Get msf record from EDQ  */
  allmsfs = allmsfs + 1             /* Up counter of all msfs   */
  msfname.allmsfs = WORD(msfedq,2)  /* Set stemmed msf variable */
end                                 /* End of loop for all MSFs */
```

OPS/REXX Instructions

For a complete discussion of REXX language instructions, see *THE REXX LANGUAGE: A* Practical *Approach to Programming* by M.F. Cowlishaw. The next few pages document differences between OPS/REXX instructions and those of standard REXX.

ADDRESS Instruction

The default ADDRESS environment is different for AOF rules (except request rules) than for REXX programs. These defaults are ADDRESS MESSAGE for all AOF rules except request rules, and ADDRESS TSO for request rules and all other REXX programs.

To change these defaults, use the CA OPS/MVS AOFDEFAULTADDRESS and REXXDEFAULTADDRESS parameters.

Important! The string following the ADDRESS keyword is treated as a constant and does not need to be enclosed in quotation marks. For example, this instruction:

```
TSO = "MESSAGE"
ADDRESS TSO
"TIME"
```

results in the host command being sent to the TSO host command environment.

If your intention is to have the host command be determined dynamically, use the following examples as a guideline, making sure to specify the host command on a separate line. Otherwise, it is ignored.

```
TSO = "MESSAGE"
ADDRESS VALUE TSO
"TIME"
TSO = "MESSAGE"
ADDRESS (TSO)
"TIME"
```

For a complete discussion of the ADDRESS instruction, see *THE REXX LANGUAGE: A Practical Approach to Programming* by M.F. Cowlishaw.

CALL Instruction

OPS/REXX supports an extension to the REXX CALL instruction.

```
CALL (name) [expression][,[expression]]...
```

The *name* is evaluated to determine the actual name of the called routine. The *name* cannot be an expression or a compound symbol itself. However, it can be a simple REXX symbol that contains the name of the routine to be called.

Because OPS/REXX resolves all external REXX subroutines at compile time, the name of any external subroutine must appear on a CALL instruction or a function call elsewhere in the program. If not, this message is issued during the compile phase:

```
REXX error 43: Routine not found
```

You may use this technique:

```
ExtRtn = 'EXTSUB3' /* For example */
CALL (ExtRtn)
/* Other REXX code */
exit
/* The following code will never be executed but is used
   to define the external subroutines used in the dynamic
   CALL above */
CALL EXTSUB1
CALL EXTSUB2
CALL EXTSUB3
CALL EXTSUB4
```

INTERPRET Instruction

OPS/REXX supports INTERPRET instructions under these conditions:

- An INTERPRET instruction can find an external function or a load module only if another instruction refers to that function or module. Most CA OPS/MVS built-in functions are dynamically located.
- When an INTERPRET instruction refers to CA OPS/MVS global variables (for example, *x* = GLOBAL.A), the global variable stem must be used directly in another instruction (not another INTERPRET instruction) elsewhere in the program. You can use global variables accessed through the OPSVALUE function in an interpreted instruction.

Note: The INTERPRET instruction executes more slowly than other REXX instructions. Therefore, use it only when system performance is not an issue or when absolutely necessary.

OPTIONS Instruction

The OPTIONS instruction under OPS/REXX accepts the following keywords:

```
OPTIONS [MAXTIME=seconds | NOMAXTIME]
  [MAXSECONDS=seconds | NOMAXSECONDS]
  [MAXCLAUSES=clauses | NOMAXCLAUSES]
  [MAXSAYS=count | NOMAXSAYS]
  [MAXCMDS=count | NOMAXCMDS]
  [MAXCOMMANDS=count | NOMAXCOMMANDS]
  [MAXSTRING=bytes | NOMAXSTRING]
  [FIRELIMIT=times | NOFIRELIMIT]
  [OPSWXTRN=name]
```

MAXTIME=*seconds* or MAXSECONDS=*seconds*

(Optional) Changes the limit on execution time.

NOMAXTIME or NOMAXSECONDS

(Optional) Skips monitoring of execution time.

MAXCLAUSES=*clauses*

(Optional) Changes the limit on clauses executed.

NOMAXCLAUSES

(Optional) Skips monitoring of clause execution.

MAXSAYS=*count*

(Optional) Changes the limit on how many SAY instructions executed.

NOMAXSAYS

(Optional) Skips limit checking for SAY instructions.

MAXCMDS=*count* or MAXCOMMANDS=*count*

(Optional) Changes the limit on host commands executed.

NOMAXCMDS or NOMAXCOMMANDS

(Optional) Skips monitoring of host command execution.

MAXSTRING=*bytes*

(Optional) Sets maximum string length for all strings.

The maximum value you can specify for the MAXSTRING keyword is 32000. The limit set by the MAXSTRING keyword is approximate; you cannot set an exact limit on string length.

NOMAXSTRING

(Optional) Uses the default maximum string length of 32000.

FIRELIMIT=*times*

(Optional) Changes how many times a rule can execute. This keyword is valid only for rules. Setting a FIRELIMIT value of zero is equivalent to setting no limit on rule execution.

NOFIRELIMIT

(Optional) Skips monitoring of rule executions.

OPSWXTRN

(Optional) Indicates to OPS/REXX which external subroutines, built-in functions, and load modules are not absolutely required to be present prior to execution. The name represents the name of an external subroutine whose presence in the environment is optional and whose absence is accounted for by the program logic.

The value of *name* must be a valid member name from 1 to 8 characters in length.

Format for OPTIONS Instructions

In an OPTIONS instruction, you must enclose all arguments except for variable names in single or double quotation marks. Enclosing arguments in quotes prevents OPS/REXX from parsing subclauses (such as MAXSAYS=5) before passing them to the OPTIONS instruction processor. CA OPS/MVS supports OPTIONS instructions in mixed case.

Duration of OPTIONS Settings

Subroutines called by a REXX program inherit the OPTIONS settings, but the settings do not apply to any calling programs. If a subroutine changes a limit or skips monitoring, the previous limit is reinstated when the subroutine returns to its caller.

If you use multiple keywords referencing the same OPTIONS setting in the same REXX statement, OPS/REXX uses the last keyword. For example, OPS/REXX will *not* limit the number of host commands if you code this instruction:

```
OPTIONS "MAXCOMMANDS=100 NOMAXCMDS"
```

Usage Notes for OPTIONS Instructions

Keep these guidelines in mind when you specify the OPTIONS instruction:

- In an OPTIONS instruction containing the OPSWXTRN keyword, the complete OPSWXTRN keyword phrase must appear on a single OPTIONS source line. In fact, we strongly recommend that OPTIONS instructions that contain OPSWXTRN keywords contain no other OPTIONS keywords.
- Because OPSWXTRN keywords are analyzed during the compile phase, no REXX substitution can be performed.
- When specifying the OPSWXTRN keyword in an OPTIONS instruction, the *name* represents the name of an external subroutine whose presence in the environment is optional and whose absence is accounted for by the program logic. The value of *name* must be a valid member name from 1 to 8 characters in length.

  Note: The validity of *name* is not checked; however, invalid member names simply waste space in an internal table.

  Some syntax errors in OPTIONS instructions that contain the OPSWXTRN keyword are flagged only during execution. See the following example.
- Regardless of whether the OPSWXTRN keyword is used, if an external REXX subroutine will ever be called at execution time, its source code must be available at compile time.
- External load modules and product built-in functions can be resolved just prior to execution and do not need to be present at compile time.
- A maximum of 24 unresolved external subroutine names are permitted in a main program or AOF rule (or any of its external subroutines). If you specify more than 24 unresolved external subroutine names, CA OPS/MVS ignores the extra ones and issues warning message OPS0924W.
- If an unresolved external subroutine is called, CA OPS/MVS issues message OPS0992E, and the REXX program terminates with error 43 to indicate that the routine was not found. The SYNTAX handler can trap this error at execution time.
- OPTIONS instructions are reprocessed during execution. Therefore, for performance reasons, it is not a good idea to place these OPTIONS instructions in a repetitive code path (for example, in a DO loop). This is true for all OPTIONS instructions, unless for some reason dynamic substitution needs to be used repetitively.
- Options that correspond to OSF server limit parameters cannot be used to selectively override the OSF server limit parameter. For example, REXX option NOMAXTIME will not override the server OSFRUN limit. The OPS/REXX options are designed to selectively override the OPS/REXX default value parameters such as REXXMAXSECONDS.

Sample Uses of OPTIONS Instructions

The following examples demonstrate the use of the OPTIONS instruction.

1. This overrides the default maximum string length for REXX programs:

```
LENGTH = 30000
OPTIONS "MAXSTRING="LENGTH
```

2. These illustrate how to limit or not limit the number of times an individual rule executes per minute:

   - To turn execution limit checking off, specify one of the following:

```
OPTIONS "NOFIRELIMIT"
OPTIONS "FIRELIMIT=0"
```

   - To execute an individual rule no more than 10 times per minute, specify:

```
OPTIONS "FIRELIMIT=10"
```

   Note: If an OPTIONS statement specifying the execution limit for a rule appears in the processing section of that rule, that limit does not take effect the first time CA OPS/MVS encounters the statement. However, the limit applies to all subsequent executions of the rule.

3. The following example, when inserted at the beginning of a REXX program executed through the OI or OX command, enables you to skip monitoring all execution limits in the program:

```
OPTIONS "NOMAXCLAUSES NOMAXTIME NOMAXSAYS NOMAXCMDS"
```

4. These illustrate valid OPTIONS instructions that use the OPSWXTRN keyword:

```
OPTIONS 'NoMaxClauses OPSWXTRN=NOTFOUND MaxSays=200'
OPTIONS "OPSWXTRN=NOTFOUND"
OPTIONS 'OPSWXTRN=NOTF'
OPTIONS 'OPSWXTRN=EXTSUB1 OPSWXTRN=EXTSUB2'
```

5. This example of the OPSWXTRN keyword used in an OPTIONS instruction illustrates a case in which CA OPS/MVS will not flag the instruction as an error but will ignore it because it is on a continuation line rather than on the OPTIONS source line:

```
OPTIONS 'NoMaxClauses MaxSays=512' || ,
        " OPSWXTRN=NOTFOU99"
```

6. These examples illustrate invalid OPTIONS instructions that use the OPSWXTRN keyword:

   - This is invalid because the value of *name* is too long. This error is flagged at run time only:

```
OPTIONS "OPSWXTRN=NOTFOUNDX"
```

   - This is invalid because the keywords in an OPTIONS instruction must be separated by blanks, not commas:

```
OPTIONS 'NoMaxClauses,OPSWXTRN=NOTF,MaxSays=200'
```

RETURN Instruction

OPS/REXX fully supports the RETURN instruction.

The)INIT,)TERM, and)PROC sections of AOF rules are OPS/REXX programs called by the AOF. The values returned by these rule sections (through the RETURN instruction) guide the actions of the AOF. For example, if a)INIT section returns REJECT, the rule is neither enabled nor executed. For details on how sections of rules execute, see the *AOF Rules User Guide*.

SIGNAL Instruction

The SIGNAL statement in OPS/REXX supports the SYNTAX operand to trap certain errors detected during execution. Because OPS/REXX uses a compile and run phase approach, true syntax errors (an invalid DO statement, for example) are detected at compile time and cannot be trapped using the SIGNAL ON SYNTAX statement. However, the SIGNAL ON SYNTAX statement can trap most run-time errors as well as global variable access or update requests that a security rule denied.

TRACE Instruction

Like standard REXX, OPS/REXX accepts the following syntax:

```
TRACE setting
```

Unlike standard REXX, OPS/REXX also supports the following syntax, which traces all changes made to the variables listed in *variables*:

```
TRACE VAR variables
```

A variable specified on the TRACE VAR instruction is ignored (not traced) if it is a:

- Global variable
- Static variable
- Repetition control variable on a DO loop

OPS/REXX supports all values for *setting* except:

!

Inhibit host command execution

?

Control interactive tracing

n* or *-n

Set interactive trace counts

S

Scan trace (without execution)

Unlike standard REXX, program lines consisting solely of comments do not appear in OPS/REXX trace output.

UPPER Instruction

The UPPER instruction translates the contents of one or more variables to uppercase. The variables are translated in sequence from left to right.

In OPS/REXX, the UPPER instruction uses this syntax:

```
UPPER variable {variable...} {;}
```

The variable is a symbol that is separated from any other variables by one or more blanks or comments. Specify only simple symbols and compound symbols.

```
a1='Hello';  b1='there'
Upper a1 b1
say a1 b1   /* Displays "HELLO THERE" */
```

If a constant symbol or stem is encountered, an error messaged is issued.

Notes:

- Any uninitialized variables will be trapped if the NOVALUE condition (SIGNAL or NOVALUE) is enabled.
- The following classes of variables cannot be used in an UPPER instruction:
  - Global variables
  - Static variables
  - AOF environmental variables

  For a definition of these classes of variables, see the *AOF Rules User Guide*.

OPS/REXX Built-in Functions

OPS/REXX offers both standard REXX functions and a set of built-in functions designed for automation. The REXX comprehensive set of built-in functions is one of its significant attractions. OPS/REXX supports all standard REXX functions as defined by the second edition of *The REXX Language: A* Practical *Approach to Programming* by M.F. Cowlishaw, plus functions specifically added for CA OPS/MVS.

For information about these built-in functions, see the chapter “OPS/REXX Built-in Functions” in the *Command and Function Reference*.

Automated Operator Facility (AOF) Global Variables

Global variables are variables that can be shared by multiple AOF rules, OPS/REXX programs, or both running in different address spaces.

Global variables are also compound symbols with any of the following stems:

- GLOBAL.
- GLOBAL*n*.
- GLVTEMP*n*.
- GLVEVENT.
- GLVJOBID.

Note: The *n* is a single digit or letter.

Use global variables as you would any other OPS/REXX variable.

The CA OPS/MVS product uses two types of global variables: *standard* and *temporary*. Standard global variables are checkpointed to data-in-virtual data sets. Temporary global variables are not checkpointed, and they do not exist across IPLs or restarts of CA OPS/MVS. The stems GLVTEMP*n*., GLVEVENT., and GLVJOBID. all identify temporary global variables.

The difference among the temporary global variable stems is the variable duration that each indicates. Consider this:

- Variables with the GLVEVENT. stem are deleted when the event for which they were created terminates.

  For example, if variable GLVEVENT.MSGTEXT is created during the processing of a message IEF405I rule, it exists for all rules that process that particular IEF405I message.

  The GLVEVENT. stem might come in handy, for instance, if you want to pass data between rules that are processing the same event, or to accumulate data of interest during the processing of the event.

  After the last rule to process the event finishes executing, the variable is deleted. The value of a GLVEVENT. variable is not shared across events-for example, the GLVEVENT.MSGTEXT variable described here will not exist when the next IEF405I message is processed by the same rules.

- A variable with the GLVJOBID. stem is deleted when the address space associated with it terminates.

  For example, if variable GLVJOBID.MSGTEXT is created during the processing of a particular job, it exists for the time during which the job is active (unless it is explicitly deleted).

As an example, you could use a GLVJOBID. variable to save data generated during one event created by the job (such as when the job issues a particular message), which could then be used to automate another action taken by the same job.

When the address space that created the GLVJOBID. variable terminates, the variable is deleted. The value of a GLVJOBID. variable is associated with only one address space; for example, references by different jobs to the GLVJOBID.MSGTEXT variable described here will have different values.

For a detailed description of uses and characteristics of both temporary and standard global variables, see the chapter “Global Variables Explained.”

Note: Because we may add other types of global variables in the future, avoid giving OPS/REXX variables stems that begin with the characters GLV.

Parsing

The following section discusses parsing.

The PARSE SOURCE Instruction

The PARSE SOURCE instruction has this format:

```
PARSE SOURCE
```

This instruction returns a string describing the source of the program being executed.

Values PARSE SOURCE Returns

The string that PARSE SOURCE returns contains these tokens:

- The characters OPS/REXX
- One of the following strings:
  - RULE, in an AOF rule
  - PROGRAM, in an OPS/REXX program
  - SUBROUTINE, in an external subroutine
- The name of the *ruleset.rulename* or program issuing the PARSE SOURCE instruction
- The name of a subroutine, if a subroutine and not a program or *ruleset.rulename* issued the PARSE SOURCE instruction

- One of the following:
  - The ddname from which the program was loaded
  - The ? character, if in a rule, in an external subroutine, or if invoked through an edit macro such as !OI
  - SYS*nnnnn* (where *nnnnn* is a numeric value), if the OX or OPSEXEC command invoked the program
- The data set name from which the program was loaded, or the ? character if a rule or a subroutine invoked the program
- The name of the initial host command environment, for instance TSO or OPER

Following are examples of strings that PARSE SOURCE returns if the source of a program is:

- An AOF rule

  ```
  OPS/REXX RULE RS1.RULE1 RS1.RULE1 ? ? OPER
  ```

- An external subroutine called by a rule

  ```
  OPS/REXX SUBROUTINE RS1.RULE1 EXSUB1 ? ? OPER
  ```

- An OPS/REXX program invoked by the OI command processor

  ```
  OPS/REXX PROGRAM PGM1 PGM1 SYSEXEC OPS/REXX TSO
  ```

- An OPS/REXX program invoked by the OX command processor

  ```
  OPS/REXX PROGRAM PGM1 PGM1 SYS00007 OPS/REXX TSO
  ```

The PARSE VERSION Instruction

The PARSE VERSION instruction has this format:

```
PARSE VERSION
```

This instruction returns a string describing the REXX language level and the release date of the REXX language processor.

Note: The REXX language level and the release date are likely to change frequently. When changed, the language level is always higher than it was in the previous release.

Values PARSE VERSION Returns

The string that PARSE VERSION returns contains these tokens:

- The characters REXX/CA
- The characters 3.92, which are a description of the REXX language level
- The characters 18 Oct 2009, which are three tokens describing the REXX language processor release date

Thus, the complete string returned by PARSE VERSION is:

```
REXX/CA 3.92 18 Oct 2009
```

OPS/REXX Interfaces

The following section discusses OPS/REXX interfaces to various components and products.

OPS/REXX Interface with ISPF Dialog Management Services

In AOF request rules and in OPS/REXX programs, you can use the ADDRESS ISPEXEC statement to pass host commands to the ISPEXEC command processor. For example, the following statement tells the ISPEXEC command processor to position the current row pointer at the top of MYTABLE:

```
address ISPEXEC
  "TBTOP MYTABLE"
```

OPS/REXX Interface with TSO

Use the ADDRESS TSO statement to pass host commands to the TSO command processor. For example, the following statement tells the OSF to submit a command for execution in an OSF TSO address space to list the attributes of data set SYS1.LINKLIB:

```
ADDRESS TSO "LISTDS 'SYS1.LINKLIB'"
```

If you execute the above instruction using the OX or OI command or through an AOF request rule executed from a TSO session, the TSO LISTDS command executes in the address space of the user. In this case, the output from the TSO command returns to the REXX program in its external data queue.

If any AOF rule other than a request rule executes the above instruction, the LISTDS command goes to a CA OPS/MVS server for execution there. Should this occur, no output returns to the AOF rule.

Support for the TSO Host Command EXECIO

The CA OPS/MVS product implements its own version of EXECIO as a TSO host command. You can use EXECIO, as documented for TSO/E REXX, in an OPS/REXX program. You *cannot* use EXECIO in a rule of *any* kind. The OPS/REXX version of EXECIO:

- Does not support the LIFO option.
- Checks the syntax of the stem name.
- Prohibits use of global variable stems with the STEM option.
- Supports DISKRU only for true sequential data sets. You cannot use DISKRU against a partitioned data set member.
- Supports mixed case variable names with the STEM option

The following OPS/REXX program segment demonstrates how you might use the EXECIO command to read and display the first 72 characters from each record in a sequential data file:

```
ADDRESS TSO
"ALLOC F(DD77) DA('SOME.DATASET.NAME') SHR REU"
IF RC <> 0 THEN
  DO
    SAY "ALLOC RC="RC
    /* Error recovery */
    EXIT
  END
"EXECIO * DISKR DD77 (FINIS"
IF RC <> 0 THEN
  DO
    SAY "EXECIO RC="RC
    /* Error recovery */
  END
DO WHILE QUEUED() > 0
  PULL REC
  SAY SUBSTR(REC,1,72)
END
"FREE F(DD77)
IF RC <> 0 THEN
  SAY "FREE FAILED; RC="RC
```

Capture TSO Command Output

The CA OPS/MVS product allocates a VIO data set to capture TSO command output. The CA OPS/MVS VIO parameter specifies the unit name to use for such VIO data sets.

Also, a known problem with the IBM TSO Terminal Monitor Program causes message IEC223I to be issued when a TSO command returns a non-zero return code. This message indicates that the VIO data set could not be closed because the TMP already freed it. Except for this extraneous message, the TMP bug has no effect on your REXX program.

OPS/REXX Interface with the OSF

Use the ADDRESS OSF statement to pass host commands to the CA OPS/MVS OSF TSO servers for execution there. For example, the following statement tells the OSF to schedule a TSO command for execution in an OSF TSO server to list the attributes of data set SYS1.LINKLIB:

```
address OSF "LISTDS 'SYS1.LINKLIB'"
```

Note: If you need information to help you decide which host environment to use (for example, to decide whether ADDRESS TSO or ADDRESS OSF better suits your purposes), see the AOF coding guidelines. The *AOF Rules User Guide* provides a summary of AOF guidelines.

Use ADDRESS OSFTSL and ADDRESS OSFTSP to schedule long-running or high priority TSO commands to OSFTSL and OSFTSP servers respectively. Use ADDRESS USS to execute USS commands in OSF USS servers.

OPS/REXX Interface with the AOF

You can use the ADDRESS AOF statement to pass host commands to the AOF. For example, the following statement tells the AOF to enable the rule named MSGRULE from a rule set named MYSET. For details, see the *AOF Rules User Guide*.

```
address AOF
  "ENABLE MYSET.MSGRULE"
```

OPS/REXX Interface with EPI

Use the ADDRESS EPI statement to pass host commands to the External Product Interface. For example, the following statement tells EPI to log virtual terminal OPSS01 to TSO:

```
address EPI
  "LOGON OPSS01 APPLID(TSO)"
```

OPS/REXX Interface to z/OS Operator Commands

Use the ADDRESS OPER statement to pass operator commands to the z/OS operating system. For example, the following statement causes z/OS to display all active address spaces:

```
address OPER
  "D A,L"
```

The ADDRESS OPER host command environment is described in the *Command and Function* Reference.

OPS/REXX Interface to Messages

Use the ADDRESS MESSAGE statement to specify an alternative environment during debugging. CA OPS/MVS customers typically use the MESSAGE host environment as the default host environment to prevent badly coded rules from issuing invalid commands, but you can specify a different default environment through the AOFDEFAULTADDRESS parameter. For more information, see the description of the AOFDEFAULTADDRESS parameter in the *Parameter Reference*.

All commands issued to ADDRESS MESSAGE generate a message (with the ID OPS4200I) displaying the command text. For example, the following statement generates the message D A,L:

```
address MESSAGE
  "D A,L"
```

You can control the disposition of such messages (and all normal CA OPS/MVS messages) by changing the message suffix.

OPS/REXX Interface to OPSCTL

Use the ADDRESS OPSCTL statement to issue commands to the CA OPS/MVS COF, ECF, MSF, and OSF components. All output from these commands goes into the external data queue, with each field separated by a blank.

The OPSCTL host environment supports the following commands:

Type of Command	Command
COF	ACTIVATE, DEACTIVATE, DEFINE, DELETE, LIST
ECF	LIST
MSF	ACTIVATE, DEACTIVATE, DEFAULT, DEFINE, DELETE, LIST, START, STOP
OSF	EXECSTATS, LIST, QUEUES, RESETQ, STOP

For more information about the OPSCTL host environment, see the *Command and Function* Reference.

OPS/REXX Interface to WTO

Use the ADDRESS WTO instruction to issue synchronous WTO messages.

For more information, see the *Command and Function Reference*.

OPS/REXX Interface to CA SYSVIEW

Use the ADDRESS SYSVIEWE statement to send commands to the CA SYSVIEW product. You can use this statement in OPS/REXX programs, but not in any type of AOF rule. The CA SYSVIEW responses to commands go into the OPS/REXX external data queue.

For a description of ADDRESS SYSVIEWE and an example of how you can use an ADDRESS SYSVIEWE statement in an OPS/REXX program, see the *Command and Function Reference*.

OPS/REXX Interface to Other CA Products Through CA GSS

OPS/REXX programs (but not AOF rules) can send commands to any CA product that interacts with the CA GSS product, including all currently supported releases of CA Jobtrac and CA Scheduler (Release 8.0 and higher). A future release of CA 7 will also support this interface. CA GSS is packaged as part of CA Common Services (CCS) for z/OS.

When an ADDRESS statement issues a command to any host environment that exists outside of CA OPS/MVS, OPS/REXX passes the command to CA GSS. CA GSS then forwards the command to the appropriate product for execution. For example, if you issue a command through an ADDRESS JOBTRAC statement, CA Jobtrac executes that command. The command output goes into the external data queue of the OPS/REXX program containing the ADDRESS statement.

When the ADDRESS statement executes, OPS/REXX sets the RC variable to one of the following:

RC has this value...	When...
-20	CA GSS fails or is not active when the program containing the ADDRESS statement executes.
-3	CA GSS is active but does not recognize the host environment.
The return code from the host command	In all other cases.

To use ADDRESS CASCHD, you must allocate the CA Scheduler Master and tracking files to the CA GSS address space. The following example shows how to send a command to CA Scheduler and retrieve the responses:

```
address CASCHD "STATUS"
SAY "RC is:" RC
do while queued() > 0
  pull line
  say line
end
```

The output should resemble the following:

```
RC is: 0
SC STATUS
 SCHEDULE JOB NAME JNO ST RC JCNT SYSID S T A T U S
 DB2DAILY                  01 0002 XAE1 STARTED
          DB31DAY4 01  40 01           WAIT START TIME 12/31 01:00
          DB31DLY4 01  40 01           WAIT START TIME 12/31 01:00
```

For more information on the CCS for z/OS components that are required to run ADDRESS CASCHD and ADDRESS JOBTRAC, see the appendix “CCS for z/OS Component Requirements” in the *Administration Guide*.

Compiler Error Messages

When the OPS/REXX compiler finds syntax errors in an OPS/REXX program or an AOF rule, the compiler generates a numbered error message. Because OPS/REXX is an implementation of standard REXX, you can find descriptions of these messages in Section 17, “Error Numbers And Messages,” of *The REXX Language: A Practical Approach to Programming* by M.F. Cowlishaw. In addition to the error codes listed there, OPS/REXX defines the following error codes:

- 91 - INVALID OR MISPLACED OPTIONS STATEMENT

  The keywords specified in the OPTIONS statement contain an error.

- 93 - GLOBAL VARIABLE WORKSPACE OVERFLOW (*size*)

  The maximum amount of storage reserved for global variables (the value set by the GLOBALMAX parameter) was exceeded.

- 94 - OVER *seconds* SECONDS USED FOR EXECUTION

  The program exceeded the maximum execution time for AOF rules (set through the AOFMAXTIME parameter) or REXX programs (set through the REXXMAXTIME parameter).

- 95 - OVER *count* HOST COMMANDS ISSUED

  The program issued the maximum number of host commands for AOF rules or REXX programs. For more information, see the descriptions of the AOFMAXCOMMANDS and REXXMAXCOMMANDS parameters in this chapter.

- 96 - OVER *count* “SAY” CLAUSES EXECUTED

  The program executed the maximum allowed SAY instructions for AOF rules or REXX programs. For more information, see the descriptions of the AOFMAXSAYS and REXXMAXSAYS parameters in this chapter.

- 97 - OVER *count* CLAUSES EXECUTED

  The program executed the maximum allowed number of clauses for AOF rules or REXX programs. For more information, see the descriptions of the AOFMAXCLAUSES and REXXMAXCLAUSES in this chapter.

The OPTIONS statement for an OPS/REXX program can also generate error codes 94 through 97.

Note: For descriptions of the parameters listed above, see the *Parameter Reference*.

More Errors Detected

Because OPS/REXX is a semi-compiler rather than a pure interpreter, its compile phase detects many errors that other versions of REXX do not catch at execution time. Especially when converting programs to OPS/REXX, you may encounter errors at compile time in supposedly error-free code. This can happen because many REXX interpreters do not detect errors in statements that do not execute.

OPS/REXX Usage Problems

The following section describes problems you may have while using OPS/REXX and ways to avoid them.

Conflicts with Internal ISPF Variable Names

Symptom:

When I use simple symbols in OPS/REXX to write ISPF dialogs, conflicts occur with the ISPF internal variables.

Solution:

When you use OPS/REXX to write ISPF dialogs, do not use simple symbols whose names begin with Z in the ISPF dialogs. Using such names causes conflicts with the ISPF internal variables.

Received Message Address Space Is Not Active

Symptom:

My program received the following message:

```
OPS3148E ADDRESS SPACE NOT ACTIVE
```

Solution:

OPS/REXX programs that update global variables or issue AOF, EPI, OPSCTL, TSO, or OPER commands require that the main CA OPS/MVS started task be active.

If you are running multiple copies of CA OPS/MVS, or you use a CA OPS/MVS subsystem name different from OPSS, you may have to either:

- Issue the ADDRESS AOF SUBSYS OPS*x* instruction to identify the correct subsystem to which subsequent ADDRESS AOF (or ADDRESS EPI, ADDRESS OPSCTL, and so in) commands will be directed, as well as subsequent global variable updates.
- Change the job step name of the TSO logon, started task, or batch job from which the REXX program is executed so that the first four characters are OPSx (the name of the correct subsystem).

If you fail to do either of the above, your program may get the message:

```
OPS3148E ADDRESS SPACE NOT ACTIVE
```

Always allocate the DD statement for the job running OPS/REXX to the subsystem where global variables reside or that will receive ADDRESS AOF, ADDRESS EPI, ADDRESS OPSCTL, ADDRESS TSO, or ADDRESS OPER commands. To do so, use JCL like that shown in the following example:

```
//OP$OPST DD DUMMY
```

The example above causes any REXX program in this subsystem to connect to subsystem OPST.

Uninitialized Variables Yield Unpredictable Results

Symptom:

My statements (usually host commands) that rely on uninitialized variable values are yielding unpredictable results.

Solution:

REXX sets any uninitialized variable to the character string that comprises the name of that variable, so you do not have to enclose literal strings in quotation marks. For example, if a variable named TSO is not initialized, REXX evaluates both TSO and "TSO" identically.

Relying on this feature to avoid using quotes can be dangerous, because if the variable is ever initialized somewhere, suddenly statements (usually host commands) that have been relying on its uninitialized value yield unpredictable results.

For instance, consider the following SAY statements:

```
say  "TSO"
SAY  TSO
```

If you want the value of the variable TSO to always be TSO, it is safer to use the SAY statement with quotation marks (say "TSO"). This coding ensures that TSO, even if initialized as a variable somewhere for another purpose, will not be evaluated, and that the literal string value will always be returned.

Problems with WTO and WTOR Messages in Subsystem Interface

Symptom:

I am finding problems related to WTO and WTOR messages in the subsystem interface.

Solution:

When you find problems related to WTO and WTOR messages in the subsystem interface, CA Customer Support may ask you to invoke a special REXX function called OPS09TRC. This function returns a string containing a z/OS control block useful for debugging purposes.

Important! Use this function only when CA Customer Support asks you to do so, because it is not intended as a programming interface and we may change it at any time.

One way to use the OPS09TRC function is to trap a WTO message through a rule and copy the returned control block strings into global variables. You can then use the OPSVIEW option 4.8 to examine the returned data.

Use the following format to call the OPS09TRC function:

```
CB = OPS09TRC("functioncode")
```

The *functioncode* argument can be any of the following values:

MAJWQE

Returns the major WQE being processed. The length of the control block returned depends on the version of z/OS you have.

MINWQE

Returns the minor WQE being processed. The length of the control block returned depends on the version of z/OS you have.

SSOB

Returns the SSOB related to the current subsystem interface call.

SSWT

Returns the WTO SSOB extension related to the current subsystem interface call.

Problems Related to Commands in Subsystem

Symptom:

I am finding problems related to commands in the subsystem interface.

Solution:

When you find problems related to commands in the subsystem interface, CA Customer Support may ask you to invoke a special REXX function called OPS0ATRC. This function returns a string containing a z/OS control block useful for debugging purposes.

Important! Use this function only when CA Customer Support asks you to do so, because it is not intended as a programming interface and we may change it at any time.

One way to use the OPS0ATRC function is to trap a particular command through a rule and copy the returned control block strings into global variables. You can then use the X line command (for hexadecimal browse) of OPSVIEW option 4.8 to examine the returned data.

A sample rule, called DEBUGCMD, demonstrates how to use the OPS0ATRC function. The DEBUGCMD rule is in the OPS.CCLXRULS data set.

Use the following format to call the OPS0ATRC function:

```
CB = OPS0ATRC("functioncode")
```

The *functioncode* argument can be any of the following values:

SSOB

Returns the SSOB related to the current subsystem interface call. Mapped by macro IEFJSSOB.

SSCM

Returns the command SSOB extension related to the current subsystem interface call. Mapped by macro IEFSSCM.

MGCR

Returns the first part of the MGCR parameter list (control portion and command buffer) related to the current subsystem interface call. Mapped by macro IEZMGCR.

Problems Related to DOM Events in Subsystem

Symptom:

I am finding problems related to DOM events in the subsystem interface.

Solution:

When you find problems related to DOM events in the subsystem interface, CA Customer Support might ask you to invoke a special REXX function called OPS0ETRC. This function returns a string containing a z/OS control block useful for debugging purposes.

Important! Use this function only when CA Customer Support asks you to do so because it is not intended as a programming interface and we may change it at any time.

One way to use the OPS0ETRC function is to trap a DOM event through a rule and copy the returned control block strings into global variables. You can then use the X line command (for hexadecimal browse) of OPSVIEW option 4.8 to examine the returned data.

Use the following format to call the OPS0ETRC function:

```
CB = OPS0ETRC("functioncode")
```

The *functioncode* argument can be any of the following values:

SSOB

Returns the SSOB related to the current subsystem interface call. Mapped by macro IEFJSSOB.

SSDM

Returns the command SSOB extension related to the current subsystem interface call. Mapped by macro IEFSSDM.

DOMC

Returns the DOM control block related to the current DOM. Mapped by macro IHADOMC.

Chapter 7: Using System State Manager

This section contains the following topics:

About SSM

The CA OPS/MVS SSM facility automates and controls the management of system resources such as started tasks, subsystems, JES initiators, and VTAM nodes.

The states of started tasks, subsystems, and other resources are always changing. They are continually started, stopped, and recycled. Often there are dependencies between resources and you cannot stop or start a resource until some other resource has been initialized or stopped. For example, TSO cannot start until VTAM initializes, and VTAM cannot be stopped while TSO remains active. Sometimes, a system ABEND or other system problem causes a started task or subsystem to terminate abnormally, requiring the operator to recover or restart it.

SSM Enhancements

SSM Version 2 is an enhanced version of the original SSM facility that provides a framework for cross-system resource management. With this framework, customers can extend their SSM applications in environments where there are resources with cross-system prerequisite relationships and resources that may run on alternative backup systems to the primary system. Specifically, SSM Version 2 includes enhancements for naming cross-system prerequisites and handling cross-system events. These enhancements allow for gathering, communicating, and maintaining information on cross-system resource relationships, and initiating actions based on this information.

SSM has also been enhanced to provide more robust prerequisite capabilities in single system and sysplex environments, exploitation of a new SQL capability to support variable length character data, new global variable syntax that simplifies defining automation actions across both sysplex and non-sysplex systems, and an audit trail for SSM table updates.

Note: The SSMVERSION parameter controls what version of SSM is started. In CA OPS/MVS Release 11.6, the SSMVERSION parameter defaults to a value of 2, and may not be set to any other value. If you are migrating to Release 11.6 from a prior release of CA OPS/MVS that was running with parameter SSMVERSION set (or defaulting) to a value of 1, you must run the OPS/REXX program OPSSM2CV to convert your SSM resource and action tables for use by SSM Version 2.

SSM Concepts

This section covers the basic concepts of SSM. SSM supervises system resources, such as running tasks and peripheral devices. The goal of the SSM engine is always to keep a resource in its desired state. When the current state differs from the desired state, SSM dispatches an action (as specified in the action text) to restore the resource to its proper state.

Understanding CURRENT and DESIRED Resource States

The terms *desired state* and *current state* refer to the contents of the DESIRED_STATE and CURRENT_STATE columns respectively for a selected resource. These columns can be set to any value that is convenient for automation needs. For example, current state can be set to INIT, STARTING, STOPPING, ACTIVE, and so on.

abstract states

SSM defines three abstract states, named UP, DOWN, and UNKNOWN. The state names that represent these states for tables are known as the table-relative UP, DOWN, and UNKNOWN states. Each resource table can have its own name for these three states; however, you can define new names to the table-relative UP and DOWN states only when adding a resource table to the SSM directory table. For example, one table of resources can have ACTIVE defined as its table-relative UP state and INACTIVE defined as its table-relative DOWN state. Another resource table can have ONLINE and OFFLINE defined as it's table-relative UP and DOWN states respectively. SSM recognizes the table-relative value by comparing the CURRENT_STATE and DESIRED_STATE columns with the UP_STATE, DOWN_STATE, and UNKNOWN_STATE columns in the directory table.

table-relative states

The state that the CURRENT_STATE and DESIRED_STATE columns are actually set to, such as ACTIVE, INACTIVE, ONLINE, and OFFLINE are table-relative states. References to the UP state refer to the table-relative state that is designated as UP. For each resource table, the value of the UP and DOWN states is defined in the directory table. The table-relative UP, DOWN, and UNKNOWN states can be referred to symbolically in action text with the corresponding column names of the SSM directory table, such as &UP_STATE, &DOWN_STATE, and &UNKNOWN_STATE.

Current state is the actual state of the resource as it appears on the system. Desired state is the state in which a resource should be. For example, if you want a subsystem to be up (desired state is UP) but it is down (current state is DOWN), the action text could be a command to start or restart the subsystem.

Note: SSM records all actions taken in response to resource state changes in OPSLOG.

Understanding Prerequisites and Subrequisites

Prerequisites are resources whose current state must be in the table-relative UP state for another resource to be started. For example, the current state of JES must be UP for TSO to be started. Prerequisites are evaluated only when the desired state of the selected resource is table-relative UP and the current state is table-relative DOWN. When this specific mismatch occurs, SSM checks the state of all prerequisites for the starting resource and defers issuing any action until all prerequisite resources are UP. For all other combinations of current and desired states, except when the desired state is DOWN and the current state is UP, SSM dispatches an action without checking any prerequisites.

Subrequisites are dependent resources that cannot be up when a prerequisite resource is down. A subrequisite is the mirror image of a prerequisite. Subrequisites are evaluated only when the desired state of the selected resource is table-relative DOWN and the current state is table-relative UP. When this specific state mismatch occurs, SSM checks the state of all resources that are subrequisite to (dependent upon) the stopping resource, and defers any action until all subrequisites are down. For all other combinations of current and desired states, except when the desired state is UP and the current state is DOWN, SSM dispatches an action without checking any subrequisites or prerequisites.

Prerequisites are named in a list (the PREREQ column) that is part of the description of each resource, kept in an RDF table. An SQL SELECT call can retrieve all the direct prerequisites of a resource from the table. However, there is no explicit subrequisite list. Instead, SSM creates a list of subrequisites for a resource, when needed, by scanning all the prerequisites for every resource in all resource tables. In the TSO/JES2 example, before SSM would terminate JES2, it would find that TSO is a subrequisite, and would add the TSO resource name to the MISSING_PREREQ column for JES2, and then take no further action on JES2 until TSO came down.

Indirect prerequisites and subrequisites are possible. For example, CICS may depend on TSO. That would make JES2 an indirect prerequisite of CICS. The searching capability of the SSM engine will find indirect prerequisites and subrequisites, even if they are indirect multiple times.

SSM uses prerequisite and subrequisite dependencies between resources to make certain that all prerequisites are satisfied before a resource is brought up, and conversely, that all subrequisite (dependent) resources are brought down before terminating their prerequisite.

You can use OPSVIEW option 4.11.2 to observe prerequisite or subrequisite conflicts from a TSO session, or use the SSMDISPC sample CMD rule to display SSM resource status from a console.

Detect State Changes for Resources

Each resource that SSM controls is represented by a row in a resource table. This row contains information about the resource, such as its current and desired states, the resource name, and the names of its prerequisite resources. Information on the current and desired states of a resource is maintained in the CURRENT_STATE and DESIRED_STATE columns respectively. An AOF message rule usually updates the CURRENT_STATE column; however, in some cases, such as during startup, it may be updated by other programs. Either a CMD rule or the Schedule Manager usually updates the DESIRED_STATE column. Both the DESIRED_STATE and CURRENT_STATE columns are monitored columns; therefore, if a change is made to either one, then the RDF posts the SSM engine to run.

When the SSM engine is posted to run, it compares the desired state of all resources to their current state and checks to see whether any missing prerequisites have been satisfied. If the current or desired state of a resource has changed, and the current state does not equal the desired state, then the SSM engine dispatches an action. The action is taken from the action text column of an action table.

In addition to scanning the resource tables when a change occurs, the SSM engine also scans all resource states every *nnnn* seconds, based on the value you set in the STATEMAXWAIT parameter. For more information about this parameter, see the chapter "Parameters for Facilities" in the *Parameter Reference.*

How SSM Works

At the heart of SSM are the Relational Data Framework (RDF) tables that store related information about each of your resources.

The RDF stores data about system resources and delivers that data to SSM through SQL queries. The data tables in the RDF keep track of the current and desired states of the system resources, and what actions to take when a resource is not in its desired state.

SSM uses the following tables to manage your system resources:

- Directory table (see page 182)
- Resource tables (see page 183)
- Action tables (see page 186)
- Auxiliary tables (see page 188)

More information:

Using the Relational Data Framework (see page 395)
Editing Relational Tables (see page 445)

Directory Table

The directory table stores the names of all resource tables and the name of the action table associated with each resource table. Action table names are stored in the same row as resource table names. SSM consults the directory table to find out which resource tables should be processed. The default name of the directory table is SSM_MANAGED_TBLS, but its name can be changed in the STATETBL parameter.

Note: For information on the STATETBL parameter, see the *Parameter Reference.*

The directory table and all of the resource tables it names are SSM managed tables. When any ADDRESS SQL function inserts or deletes a row from a managed table, the SSM engine immediately wakes up and reevaluates its missing prerequisites, subrequisites, and resource states. In addition, the SSM engine designates some columns as monitored columns. When an SQL UPDATE state affects a monitored column, the SSM engine wakes up and scans the same data.

The following shows a directory table as viewed in the CA OPS/MVS table editor. When column definitions extend past the right margin of your terminal screen, issue RIGHT commands or press the PF11 key to display more of the data.

```
SSM_MANAGED_TBLS ---------------------- TABLE DATA EDITOR ------------------ COLUMNS 000
Command ===>                                                  Scroll =
MANAGED_TABLE  TABLE_MODE  UP_STATE  DOWN_STATE  UNKNOWN_STATE  ACTION_TABLE  TNGELIGIBLE
*******************************************************************************************
SWZ_STCTBL      ACTIVE      UP        DOWN        UNKNOWN        STCTBL_ACT     YES
********************************** BOTTOM OF DATA ******************** ******************
```

The descriptions of the columns in the directory table are as follows:

MANAGED_TABLE

Identifies the name of a resource table SSM should manage.

TABLE_MODE

Indicates the operating mode of the resource table. This mode determines whether SSM checks the state of the prerequisites for each resource and tries to manage the resource.

Values: ACTIVE, INACTIVE, NOPREREQ, and PASSIVE

UP_STATE

Indicates the table-relative UP state (such as ACTIVE or UP) for a resource.

DOWN_STATE

Indicates the table-relative DOWN state (such as INACTIVE or DOWN) for a resource.

UNKNOWN_STATE

Indicates the table-relative state for the resource when SSM is to determine the state. This column is usually set to UNKNOWN, but you can change it.

ACTION_TABLE

Identifies the action table associated with the table specified in the MANAGED_TABLE column.

TNGELIGIBLE

(Optional) Displays a YES or NO value to indicate whether a resource table is eligible for display by the CA Network and Systems Management System Status Manager CA OPS/MVS Option product.

The resource directory table cannot contain more than 256 managed tables. If the 256 limit is exceeded, the error message OPS7911E is generated and the SSM mode changes to INACTIVE.

Resource Tables

Resource tables store information about the status of each resource. Although you could place data on all of your system resources in a single table, we recommend that you create a separate resource table for each type of resource; for example, a table for VTAM nodes, another table describing printers, and so on.

The following resource table named STCTBL shows a subsystem resource table as viewed in the CA OPS/MVS table editor. When column definitions extend past the right margin of your terminal screen, issue RIGHT commands or press the PF11 key to display more of the columnar data.

```
STCTBL ------------------------------------------------ TABLE DATA EDITOR ------------------------------- COLUMNS 00001 00124
Command ===>                                                                                                Scroll ===> PAGE
COL--> NAME              CURRENT_STATE DESIRED_STATE MODE     PREMODE  REFMODE  ACTMODE  SCHEDMODE JOBNAME  TYPE        CHK
****** ***************************************************** TOP OF DATA ***********************************************************
000001 APPC              UP            UP            ACTIVE   ACTIVE   ACTIVE   ACTIVE   ACTIVE    APPC     APPC        UNK
000002 ASCH              UP            UP            ACTIVE   ACTIVE   ACTIVE   ACTIVE   ACTIVE    ASCH     APPC        UNK
000003 ASM2              UP            UP            ACTIVE   ACTIVE   ACTIVE   ACTIVE   ACTIVE    IXRASUBS ASM2        UNK
000004 BALFSG            UP            UP            ACTIVE   ACTIVE   ACTIVE   ACTIVE   ACTIVE    BALFSG   UNKNOWN     UNK
000005 BBOASR2           UP            UP            ACTIVE   ACTIVE   ACTIVE   ACTIVE   ACTIVE    BBOASR2  UNKNOWN     UNK
000006 BBODMN            UP            UP            ACTIVE   ACTIVE   ACTIVE   ACTIVE   ACTIVE    BBODMN   UNKNOWN     UNK
000011 MIMGX             DOWN          UP            ACTIVE   ACTIVE   ACTIVE   ACTIVE   ACTIVE    MIMGX    UNKNOWN     UNK
000007 NBOIR             UP            UP            ACTIVE   ACTIVE   ACTIVE   ACTIVE   ACTIVE    BBOIR    UNKNOWN     UNK
000008 NBOLDAP           UP            UP            ACTIVE   ACTIVE   ACTIVE   ACTIVE   ACTIVE    BBOLDAP  UNKNOWN     UNK
000009 PBONM             UP            UP            ACTIVE   ACTIVE   ACTIVE   ACTIVE   ACTIVE    BBONM    UNKNOWN     UNK
000010 PBOSMS            UP            UP            ACTIVE   ACTIVE   ACTIVE   ACTIVE   ACTIVE    BBOSMS   UNKNOWN     UNK
```

The descriptions of the columns in the STCTBL resource table are as follows:

NAME

Identifies the name of the resource, such as JES2 or VTAM. This resource name is usually the job name of the subsystem.

CURRENT_STATE

Indicates the actual state of the subsystem.

DESIRED_STATE

Indicates the preferred state of the subsystem.

MODE

Indicates the way in which SSM will monitor and control the subsystem.

PREMODE

Specifies a setting that controls whether prerequisites and subrequisites are processed for the resource when the desired state of a resource is UP and the current state is DOWN, or the desired state is DOWN and the current state is UP.

Note: Setting both PREMODE='INACTIVE' and REFMODE='INACTIVE' disconnects the current resource from any prerequisite or subrequisite processing.

REFMODE

Specifies a setting that controls whether the current resource permits other resources to reference the state of the current resource for purposes of prerequisite and subrequisite processing.

Note: Setting both PREMODE='INACTIVE' and REFMODE='INACTIVE' disconnects the current resource from any prerequisite or subrequisite processing.

ACTMODE

Displays an additional mode value that can temporarily override the resource processing mode or be used to execute different actions for the same state combinations. A change in the value of the ACTMODE column of a resource causes the resource to be selected for processing, regardless of the values of its current and desired state.

The ACTMODE column is compared to the ACTION_MODE column in the action table during the select action procedure.

Note: For more information on the select action procedure, see How SSM Decides What Action to Take (see page 213) in this chapter.

SCHED_MODE

(Optional) If the value of this column is INACTIVE, then Schedule Manager RESET processing bypasses any updates to the DESIRED_STATE column of this resource.

Note: For more information on this schedule override feature, see the chapter "Using Schedule Manager (see page 315)."

JOBNAME

Specifies the job name for the subsystem.

TYPE

Identifies the type of subsystem, for example, IMS, CICS, or VTAM.

CHKPOINT_STATE

Same as the value for DESIRED_STATE. The CHKPOINT_STATE column provides one way to reset desired resource states after a restart of CA OPS/MVS.

IPL_STATE

Displays the state in which the subsystem should be following an IPL.

TNGNOTIFY

(Optional) Determines whether the resource is displayed as an icon on the CA map created by the CA Event Manager product to monitor SSM resources.

RESOURCE_TEXT

(Optional) Contains documentation for the SSM resource. This information is displayed on the CA Network and System Management System Status Manager CA OPS/MVS Option workstation. You could use this column to display the text of the last z/OS message that has changed the state of the resource.

PREREQ

Displays a list of resources that must be in an UP state before SSM can start the subsystem. Maximum length is 1000 characters.

MISSING_PREREQ

Displays a list of the prerequisite resources that have not been satisfied. Maximum length is 1000 characters. A VARCHAR definition is used to conserve memory.

PREV_STATE

Tells SSM whether the state of a subsystem has changed.

INTERNAL_DATA1

Contains data that SSM uses internally to control the subsystem.

PRIMARY_SYSTEM

(Optional) If present, the contents of this column sets the variable SSM!PRISYS.

More information:

Resource and Action Tables (see page 187)

Action Tables

Columns in action tables determine the action SSM takes to change a resource from its current state to its desired state. When the current and desired states of a resource do not match (for instance, if the desired state is UP and the current state is DOWN), SSM searches the action table associated with the resource table defining that resource, finds the action required to perform the state change, and executes that action.

The following shows a subsystem action table as viewed in the CA OPS/MVS table editor. When column definitions extend past the right margin of your terminal screen, issue RIGHT commands or press the PF11 key to display more of the columnar data.

```
STCTBL_ACT ---------------------------------------- TABLE DATA EDITOR ------------------------------- COLUMNS 00001 00124
Command ===>                                                                                          Scroll ===> PAGE
COL--> ACTION_PROCESS ACTION_CURRENT ACTION_DESIRED ACTION_MODE ACTION_RES_TABLE ACTION_RES_TYPE ACTION_TEXT
****** ************************************************ TOP OF DATA ************************************************************
000001 ACTION         DOWN           UP                                                        MVSCMD("START &JOBNAME")
000002 ACTION         DOWN           UP             MONITOR                                    MVSCMD("D A,&JOBNAME")
000003 ACTION         UNKNOWN                                                                  RULE("SSMSTATE
000004 ACTION         UP             DOWN                                                      MVSCMD("STOP &JOBNAME")
000005 ACTION         UP             DOWN           MONITOR                                    MVSCMD("D A,&JOBNAME")
000006 ACTION         DOWN           DOWN           CYCLE                                      SETCOL("ACTMODE,ACTIVE")
000007 ACTION         UP             UP             CYCLE                                      SETCOL("DESIRED_STATE,&DOWN_STATE")
000008 SELECT                                                      STCTBL                      PROCESS("MATCH,XPREREQ")
000009 SELECT         UNKNOWN                       ACTIVE         STCTBL                      PROCESS("MAT,XPRE,XSUB")
000010 XPREREQ                                                                                 RULE("SSMXPRE &NAME")
000011 XSUBREQ                                                                                 RULE("SSMXPRE &NAME")
****** ********************************************** BOTTOM OF DATA ************************************************************
```

The descriptions of the columns in the STCTBL action table are as follows:

ACTION_PROCESS

Displays the primary selection criterion SSM uses when selecting an action to dispatch. To be eligible for selection, ACTION_PROCESS must match the process event for the action. The event name for selecting an action to be dispatched in response to a resource state mismatch is ACTION. Other possible values are MPREREQ, MSUBREQ, and the other process event types listed in Process Events in this chapter.

ACTION_CURRENT

Displays a particular CURRENT state in which the subsystem resides.

ACTION_DESIRED

Displays a particular DESIRED state that the subsystem can be in.

ACTION_MODE

Displays an additional action selection key that corresponds to the ACTMODE column of the resource table. An empty string value in this column (the default value) causes this column to be ignored in action selection.

ACTION_RES_TABLE

Displays the name of the resource table that defines the subsystem to SSM. This column enables one action table to supply actions for the resources in more than one resource table.

ACTION_RES_TYPE

Displays either the name or the type of the subsystem as defined in the NAME or TYPE columns in the resource table. This column enables SSM to act upon a specific resource.

ACTION_TEXT

Indicates the action to take for a state change. This text is either the text of a TSO command or the name of an automation procedure that can restore the resource to its desired state.

More information:

Resource and Action Tables (see page 187)

Resource and Action Tables

Resource tables contain a list of resources and information about these resources, such as names, types, prerequisites, current and desired states, and previous states. The RDF may contain many different resource tables.

Each resource table is associated with one action table. Action tables contain action clauses that execute when a resource is not in its desired state. They tell SSM what to do to manage the resources. An action table can be associated with many resource tables, but each resource table can be associated with only one action table.

For an example of how resource and action tables work together, see the sample resource table (see page 183) and action table (see page 186) in this chapter. These sample resource and action tables show that MIMGX has a current state value of DOWN and a desired state value of UP. SSM reads the action table and finds the row where the ACTION_PROCESS column value equals ACTION, the ACTION_CURRENT column value (DOWN) matches the CURRENT_STATE column value (also DOWN), and the ACTION_DESIRED column value (UP) matches the UP value in the DESIRED_STATE column.

Since none of the ACTION_RES_TYPE columns for DOWN to UP match the resource name or type of MIMGX, the default DOWN to UP action table row (the first row) is selected. This matching causes SSM to perform the action in the ACTION_TEXT column of this row, which in this case issues a z/OS command to restart job MIMGX.

Auxiliary Tables

Auxiliary tables should be used as follows:

- To store status information about resources that reside outside of the local system, such as the status of started tasks on an external system

  The SSM engine can reference these statuses as needed without the complication and delay imposed by communicating with another system.

- When SSM requires the status of many external resources

  If you write an application that uses an auxiliary table, then you need to supply a task to manage communications and keep the table current. Using auxiliary tables in situations where SSM references only a low volume of external statuses is not justified. Changes to auxiliary tables cause the SSM engine to reevaluate the status of known resources, but do not provide any information about how the auxiliary table was changed.

- To implement a global control and response feature under SSM
- To track status changes to non-local resources

Sample programs are available that make use of auxiliary tables; however, they are for example purposes only. We do not guarantee correct or useful results from tables created by these sample programs.

Note: There are no required columns in auxiliary tables and they are not listed in the directory table.

Important! Auxiliary tables should be used only for SSM-related activity because changes to these tables wake up SSM and cause it to check the status of all local resources. For this reason, overusing auxiliary tables creates unnecessary overhead.

The only way SSM can distinguish an auxiliary table from other tables is by its name. If the prefix of a table name matches the prefix defined in the SSMAUXTBLPREFIX parameter, then the table is an auxiliary table. This prefix can be one to six characters long.

When an auxiliary table is updated (except by the SSM engine), for example by adding, deleting, or updating a row, the SSM engine is posted, causing the engine to reevaluate all missing prerequisites. Updates to auxiliary tables by the SSM engine do not cause the engine to post. Auxiliary tables are loosely coupled with SSM since no data is transmitted to the SSM engine when activity wakens the engine. An action, executed by the SSM engine, can examine the contents of the SSM auxiliary tables at any time. Auxiliary tables are not SSM tables, and do not produce SSM global event activity. Specifically, ADD, DELETE, and UPDATE events are not generated for auxiliary tables.

SSM Resource Management Modes

SSM can operate in four resource management modes, ACTIVE, NOPREREQ, PASSIVE, and INACTIVE. These modes define how SSM responds to state changes in resources that it controls. Mode can be set individually for each resource, for all resources in a table, or for all resources in all tables on the local system. When settings conflict, the more restrictive setting applies. For example, if a table is set to PASSIVE mode and a resource in the table is set to ACTIVE mode, then the resources are managed in PASSIVE mode.

The following list describes the four management modes listed in order of increasing restrictiveness:

- ACTIVE

  SSM functions fully, managing resources and tables and taking actions when the desired state of a resource differs from its current state. SSM can start and stop resources while in ACTIVE mode.

- NOPREREQ

  SSM operates as it does in ACTIVE mode, except that it bypasses checking prerequisites or subrequisites. NOPREREQ is most often applied to an individual resource, but it can be applied to a table or an entire system if needed. An EMERGENCY shutdown procedure can set NOPREREQ mode for a table or system to quickly terminate resources.

- PASSIVE

  SSM does not trigger any action for deviations between current state and desired state except when the current state is equal to the table-relative UNKNOWN state. If the current state is equal to the table-relative UP or DOWN states, then desired state is set to the current state. Otherwise, the desired state is set to the table-relative UNKNOWN state.

  You can use MODE=PASSIVE when you are not sure what action SSM will take but you want to synchronize the current state and desired state values of managed resources with their actual state on your system. You can also use MODE=PASSIVE when you do not want to take control of a resource but want SSM to track its current state.

- INACTIVE

  SSM does not react to deviation between the current state value and the desired state value for a resource.

  Important! When a prerequisite resource is set to INACTIVE mode, it still affects the ability of its subrequisites to run.

Define Resource Management Modes for SSM

Mode can be set for all SSM resources, all resources in a table, or per resource. The mode you specify for the STATEMAN parameter causes SSM to use that mode to manage all of its tables and resources. For example, if you set STATEMAN=PASSIVE, all tables listed in the directory table and all the resources defined in those tables will be monitored in PASSIVE mode.

Similarly, if you set STATEMAN=ACTIVE and want to use INACTIVE mode for one table without affecting the other tables, you can do so by setting the TABLE_MODE value for that table to INACTIVE in the directory table's definition of that table. You can also set the management mode for an individual resource by setting the MODE column in the resource table's definition of that resource. In the hierarchy of mode values, the most restrictive mode always applies.

Some aspects of resource management mode are also affected by settings other than resource management mode. For example, the PREMODE and REFMODE columns of the resource tables affect prerequisite and subrequisite processing.

The mode SSM uses to monitor resources directly affects what values the CURRENT_STATE and DESIRED_STATE columns of your resource tables contain. If CA OPS/MVS becomes active, both the CURRENT_STATE and DESIRED_STATE columns for a resource will be set to the value in the table-relative UNKNOWN state for that resource as defined in the directory table. The same is true if SSM is running in INACTIVE mode and you change the mode to ACTIVE or PASSIVE, or if you set the monitoring mode for a specific table using the OPSSMTBL command.

To take full advantage of operating in PASSIVE or ACTIVE mode, you need to prime the DESIRED_STATE column. For an explanation of how to do this, see Setting the Desired State in this chapter.

To define resource management modes for SSM

- Set the processing mode of a system, table, or resource using OPSVIEW option 4.11.1 (System State Manager Control) or option 4.11.2 (System State Manager Resource Control).

  You can also use the following methods:

- To set the SSM monitoring globally

  Set the STATEMAN parameter by editing the CA OPS/MVS parameter file, by using OPSVIEW option 4.1.1, or by using the OPS/REXX OPSPRM function or the OPSPARM TSO command:

  ```
  xxx=OPSPRM('SET','STATEMAN','ACTIVE')
  OPSPARM SET(STATEMAN) VALUE(ACTIVE)
  ```

- To set the mode for a particular table

  Issue the OPSSMTBL command processor as follows:

```
OPSSMTBL CHANGE(tablename) MODE(modevalue)
```

- To set the SSM management mode for an individual resource

  Use the MODE column in the resource table:

  ```
  OI STATESET table.name MODE(newmode)
  ```

  MODE

  The MODE column in a resource table can be used to set the resource management mode of an individual resource to a more restrictive level than the level set by the STATEMAN parameter or the value of the MODE column in the directory table; however, it cannot override a more restrictive general setting. For example, if the management mode of a table is set to ACTIVE and the STATEMAN parameter is set to PASSIVE, then setting the value of the MODE column to ACTIVE for a resource would result in an effective mode of PASSIVE. In the same situation, setting the value of the MODE column of a resource to INACTIVE would make the resource management mode of that resource INACTIVE.

Prerequisites

As stated earlier in the chapter, prerequisites are resources whose current state must be in the table-relative UP state for another resource to be started. The following sections describe how prerequisites work and how to define and control them.

Check the State of Prerequisite Resources

Prerequisite checking occurs when the desired or current state of a resource changes, and the resulting desired state is UP and the resulting current state is DOWN. Subrequisite checking occurs when the desired or current state of a resource changes, and the resulting desired state is DOWN and the resulting current state is UP. The terms UP and DOWN refer to table-relative up and down states, which are defined in the directory table. Prerequisite and subrequisite checking do not occur if the desired and current states match each other (that is, both are UP or both are DOWN), or if either state does not match one of the table-relative UP or DOWN states. Table-relative states are considered prefixes when compared to the desired state and current state to determine a match. For example, a desired state of UPWAIT matches a table-relative state of UP because UP is considered a prefix and it matches the prefix UP in UPWAIT.

Desired Table States

A resource table has a table-relative UP state of RUN, and a table-relative DOWN state of STOP. The desired state is RUNPEND and the current state is STOPPED. This is considered to be a DOWN to UP transition because the first three characters of desired state RUNPEND match the table-relative UP state of RUN, and the first four characters of current state STOPPED match the table-relative DOWN state. This transition will match the STOPPED/RUNPEND entry in the action table, and prerequisite checking will also be done.

Consider a VTAM resource table, for which the table-relative UP state is defined as ACTIVE and the table-relative DOWN state is defined as INACTIVE. If the desired state changes to ACTIVE or the current state is INACTIVE, then prerequisite checking occurs. Likewise, if the desired state changes to INACTIVE and the current state is ACTIVE, then subrequisite checking occurs. If, however, the desired state changes to ACTIVE and the current state changes to STARTING, then neither prerequisite nor subrequisite checking occurs; however, an action will still be asserted.

For example, suppose that you have a directory table containing the following columns and values:

MANAGED_ TABLE	TABLE_ MODE	UP_STATE	DOWN_STATE	UNKNOWN_ STATE	ACTION_TABLE
STCTBL	ACTIVE	UP	DOWN	UNKNOWN	STCTBL_ACTION
VTAM_NODES	ACTIVE	ACTIVE	INACTIVE	UNKNOWN	VTAM_ACTION

Also suppose that you have the following entries in the STCTBL resource table (not all columns shown):

NAME	DESIRED_ STATE	CURRENT_ STATE	PREREQ	MISSING_PREREQ
JES2	UP	UP	NULL	NULL
VTAM	UP	STARTING	JES2	NULL
PRODCICSA	UP	DOWN	VTAM, VTAM_NODES.CICSA	VTAM, VTAM_NODES.CICSA

Finally, suppose that you have the following entries in the VTAM_NODES resource table (not all columns shown):

NAME	DESIRED_STATE	CURRENT_ STATE	PREREQ	MISSING_ PREREQ
CICSA	INACTIVE	INACTIVE	NULL	NULL

The resource PRODCICSA in the STCTBL resource table has a desired state value of UP, which is defined as its table-relative UP state in the directory table. However, SSM will not execute the automation procedure to set the current state of PRODCICSA to UP because the prerequisites for PRODCICSA are not in their defined UP states. As you can see in the examples, PRODCICSA has prerequisites of VTAM, whose current state is STARTING, and CICSA in the VTAM_NODES table, whose current state is INACTIVE.

Note: These unsatisfied prerequisites are also listed in the MISSING_PREREQ column of the STCTBL resource table.

Once the current state of VTAM is set to UP and the current state of CICSA is set to ACTIVE, SSM can take the action needed to place PRODCICSA in its defined UP_STATE.

A resource that is a prerequisite for another resource may exist in any SSM table. When a resource and its prerequisites are defined in different monitored tables, the prerequisite resource must be defined as *tablename.resourcename*, as shown with VTAM_NODES.CICSA in the preceding examples.

The Effect of STATEMATCHPREFIX on Prerequisite Checking

Once SSM determines that a prerequisite check is needed, it compares the current state of each prerequisite with the table-relative UP state for the table in which the prerequisite is stored.

The STATEMATCHPREFIX parameter can be set to NO to indicate that resources should be considered UP for prerequisite purposes only if their current state exactly matches the table-relative up state.

If the STATEMATCHPREFIX parameter is set to YES, then the prerequisite is considered to be UP if the table-relative UP state, treated as a prefix, matches the current state of the prerequisite.

This allows a started task that is in an UPBEGIN state to be considered UP for prerequisite purposes.

The effect of this parameter on positive and negative prerequisites and subrequisites is the same as that of normal prerequisites.

More information:

Define Positive and Negative Prerequisite Resources (see page 196)

Define Prerequisite Resources

This section discusses naming prerequisite resources, the various methods you can use to define prerequisite resources, and the types of prerequisite resources that can be defined.

Specify the Name of Prerequisite Resources

The automation programmer has the option of specifying a system name as part of a prerequisite resource name. The system name is parsed by SSM and passed to process action code as an environmental variable. SSM processes the prerequisite as local if any of the following occurs:

- The system name is omitted.
- The XPREREQ process action is not enabled. For more information, see Process Events in this chapter.
- The system name is specified as a single asterisk (*).

To specify a system name as part of a prerequisite name, use the following syntax:

`[[[`*`system name`*`.][`*`ssid`*`.]`*`table`*`.]`*`resource`*

system name

1 to 9 characters in length. There are two special cases for prerequisite names that have a non-blank system name:

- If the value of *system name* is an asterisk (*), then the prerequisite is on the local system
- For a Work Load Manager (WLM) resource request, if *system name* is specified, the value of *ssid* is WLM, and the value of *table* is *SCHENV, then the WLM API evaluates the resource

Both of the above are handled by SSM without asserting an XPREREQ process event. For all other cases when *system name* is non-blank, SSM asserts an XPREREQ event to process the name. However, if XPREREQ is not enabled, then SSM processes the prerequisite as if it were local.

Customers are encouraged to use *system name* to implement their own prerequisite types, such as a system affinity requirement or an ARM element name status. However, to avoid future conflicts over naming conventions, an exclamation point (!) should be used as the first character of any system name. The remaining 8 characters of the name can be any character string. System names starting with any character other than an exclamation point may cause unpredictable results in future releases of CA OPS/MVS.

When a system other than local or WLM is specified, the automation programmer is responsible for providing a REXX program that obtains any cross-system status needed.

Important! If you designate a system name, then you must provide an XPREREQ action and enable the event. If this action is not present and enabled, then the SSM engine will process the specification as if it were local.

ssid

1 to 4 characters in length. *ssid* cannot be specified unless *system name* is specified. If only one character is specified, then the prefix OPS is added to it to form the subsystem name. If *ssid* is blank or an asterisk (*), then the name of the local CA OPS/MVS SSID is used.

Note: An SSID name of WLM is reserved for use by WLM-related resources only.

table

1 to 18 characters in length. The name of the RDF table that contains the resource description.

resource

1 to 18 characters in length. The name of the resource.

MINOF Statement—Define Prerequisite Resources

In the previous example, all of the resources in the prerequisite list had to be in their table-relative UP state before SSM could dispatch an action to cause the desired state of a resource to get to its table-relative UP state or DOWN state. With the MINOF (minimum of) statement, you define how many prerequisite resources must be in their table-relative UP state before a resource can be brought to its desired state.

For example, suppose you have the following MINOF statement:

```
MINOF(1,VTAM,VTAMTEST)
```

1

Defines the minimum number of prerequisite resources that must be in their table-relative UP state

VTAM and VTAMTEST

Defines the prerequisite resources.

In the above statement, VTAM, VTAMTEST, or both can be in their table-relative UP state for SSM to take the action needed to place the resource in its table-relative UP state.

Following is additional information about the MINOF statement:

- The MINOF statement can be combined with other prerequisites and with other MINOF statements. Also, MINOF statements can be nested in MINOF statements; however, do not nest them more than three levels. Following is an example of a MINOF statement combined with other resources:

  ```
  MINOF(1,VTAM,VTAMTEST),JES2
  ```

  Following is an example of a nested MINOF statement:

  ```
  MINOF(1,MONTH_END,MINOF(1,DAY_SUMRY,DAY_TOTALS))
  ```

- The MINOF statement is evaluated for subrequisite processing as well as for prerequisite processing. Subrequisites are found by searching prerequisites for other resources. When a MINOF statement is found in a subrequisite resource, the state of the subrequisite resource (UP or DOWN) is evaluated as if the state of the current resource, for which subrequisites are being evaluated, was DOWN.

Define Positive and Negative Prerequisite Resources

Any resource in a prerequisite list can be qualified as a positive or negative prerequisite by adding a plus (+) or minus (-) sign before its name. A positive prerequisite resource is considered to be satisfied when the specified resource is UP, and a negative prerequisite resource is considered to be satisfied when the specified resource is DOWN. Note

Note: Prerequisites that have been prefixed with a plus or minus are never evaluated as subrequisites.

A plus or minus prefix can be applied to a MINOF statement, to resources in a MINOF statement, and to individual resources.

Making two resources negative prerequisites of each other creates a mutually exclusive relationship. For example:

```
VTAMPROD PREREQ(-VTAMTEST)

VTAMTEST PREREQ(-VTAMPROD)
```

Define Positive and Negative Subrequisite Resources

The positive and negative subrequisite capability is the equivalent of the positive and negative prerequisite capability described above.

To define positive and negative subrequisite resources

1. You can qualify the resources in a prerequisite list as a positive or negative subrequisite by adding a less than (<) or a greater than (>) sign before its name.
   - A positive subrequisite is considered to be satisfied when the resource with the subrequisite is DOWN.
   - A negative subrequisite is considered to be satisfied when the resource with the subrequisite is UP.

Notes:

- Resources in a prerequisite list that have been prefixed with a less than or greater than character are never evaluated as prerequisites. They are only evaluated as a subrequisite to the resource in the NAME column.
- To avoid over complexity, the positive and negative subrequisite capability can not be utilized within a MINOF statement.

Example: Positive and Negative Subrequisite Resources Definitions

Suppose that you are managing two resources; TASK1 and TASK2. Although you are not concerned in which order they start, you want to make sure that TASK2 is always stopped before TASK1. This is an example of when a positive subrequisite would be useful. The table would look like this:

NAME	DESIRED_STATE	CURRENT_STATE	PREREQ
TASK1	UP	UP	
TASK2	UP	UP	>TASK1

In this table, TASK1 is listed as a positive subrequisite for TASK2. When System State Manager goes to start TASK2, it ignores the >TASK1 value in the PREREQ column. Furthermore, TASK1's definition as a positive subrequisite for TASK2 has no influence on System State Manager's starting process for TASK1.

The result is that either resource can be started first. However, any attempt to stop TASK1 first before stopping TASK2 will fail because TASK1 is a subrequisite for TASK2.

In a variation of this example, suppose you are not concerned in which order they start, but you wanted to make sure TASK1 could only be stopped when TASK2 was UP. This could be achieved by making TASK1 a negative subrequisite for TASK2:

NAME	DESIRED_STATE	CURRENT_STATE	PREREQ
TASK1	DOWN	DOWN	
TASK2	UP	UP	<TASK1

In this table, TASK1 is listed as a negative subrequisite for TASK2. When System State Manager goes to start TASK2, it ignores the <TASK1 value in the PREREQ column. Furthermore, TASK1's definition as a negative subrequisite for TASK2 has no influence on System State Manager's starting process for TASK1. However, any attempt to stop TASK1 will only be honored if TASK2 is UP. Attempts to stop TASK1 will fail if TASK2 is already DOWN.

Define a Workload Manager Scheduling Environment as a Prerequisite Resource

SSM can define the status of a Workload Manager (WLM) scheduling environment as a prerequisite resource. A special naming convention must be used to specify the status of a WLM environment as a resource. The format is:

```
System.WLM.*SCHENV.Name
```

System

Specifies a valid z/OS system name that is known to WLM

Name

Specifies the WLM scheduling environment name. *Name* can be up to 16 characters in length.

WLM

Provides a placeholder that must be used in the SSID position for WLM resources if *System* is specified.

To access the status of only the local WLM, specify the following:

```
*SCHENV.Name
```

Notes:

- The WLM scheduling environment is the only cross-system resource that SSM evaluates without asserting a process event.
- The status of a WLM scheduling environment cannot be used as a subrequisite resource.
- The status of a WLM scheduling environment can be used by SSM to recognize an abstract condition such as Prime Shift or Batch Window.

The ability of SSM to read the status of a WLM scheduling environment eliminates the need for it to compute the status itself. This increases the efficiency of SSM and avoids the possibility that SSM will compute a WLM status differently than WLM itself because of sample timing differences.

The OPSWLM Function

OPSWLM is an OPS/REXX function that can be used to set the status of a WLM scheduling environment resource. This function allows you to indirectly control the status of a scheduling environment. The ability to set the status of a WLM scheduling environment resource can be used in conjunction with the ability to synchronize CA OPS/MVS with WLM.

For more information, see OPSWLM Function in the chapter “OPS/REXX Built-in Functions” in the *Command and Function Reference*.

Control Prerequisite Resources

The PREMODE, REFMODE, and MODE columns can be used to control prerequisite processing. This section discusses each column in detail.

PREMODE Column

The PREMODE column in the resource table controls whether prerequisite and subrequisite processing is performed for a resource, when DOWN to UP or UP to DOWN transitions occur. Valid values for this column are:

ACTIVE

Both prerequisite and subrequisite processing is performed according to the standard method used by SSM.

INACTIVE

Neither prerequisite nor subrequisite processing is performed. This is equivalent to the SSM NOPREREQ mode.

PREREQ

Only prerequisite processing is performed for this resource. Subrequisite processing is bypassed.

SUBREQ

Only subrequisite processing is performed for this resource. Prerequisite processing is bypassed.

REFMODE Column

The REFMODE column controls whether other resources should process this resource when testing their prerequisite or subrequisite references.

Valid values for this column are:

ACTIVE

Both prerequisite and subrequisite references are performed according to the standard method used by SSM.

INACTIVE

Neither prerequisite nor subrequisite references to this resource are evaluated. This is equivalent to the SSM NOPREREQ mode. When REFMODE INACTIVE is combined with PREMODE INACTIVE, this resource is disconnected from all prerequisite and subrequisite processing.

PREREQ

Only prerequisite processing references to this resource are evaluated. Subrequisite references to this resource are ignored.

SUBREQ

Only subrequisite processing references to this resource are evaluated. Prerequisite references to this resource are ignored.

MODE Column

The MODE column in a resource table controls the resource management mode of an individual resource. When the MODE column is set to NOPREREQ, SSM operates as it does in ACTIVE mode, except that it bypasses checking prerequisites or subrequisites.

More information:

Define Resource Management Modes for SSM (see page 190)

Initializing Data

When SSM becomes active, it sets the CURRENT_STATE and DESIRED_STATE columns to the table-relative UNKNOWN state. An automation procedure must set the desired state values or no action will occur. The SSMBEGIN request rule is provided for this purpose. SSMBEGIN is automatically invoked at SSM initialization and whenever the mode (STATEMAN parameter) is upgraded from INACTIVE to PASSIVE or ACTIVE, or from PASSIVE to ACTIVE.

Methods for Setting the Desired State

SSM provides several methods, described below, for initializing the desired states for resources. The method or methods chosen depend on the state that CA OPS/MVS is in, the type of resource, and how your system manages the resource.

These methods are:

- Manually starting and stopping resources
- CA OPS/MVS TOD rules
- The CA OPS/MVS Schedule Manager feature
- Checkpointing resources
- The CA OPS/MVS OPSVIEW feature
- The CA OPS/MVS SSMBEGIN request rule
- A method you define

Set the Desired State Manually

Manually starting and stopping system resources that are being actively controlled by SSM is a common daily activity within any z/OS environment, and can be performed using the following methods:

- The standard z/OS START and STOP commands
- A pseudo command to trigger a CMD rule to control SSM activity
- The supplied STATESET OPS/REXX program

Use z/OS START and STOP Commands

With SSM actively monitoring the resources that are being manually requested to START or STOP, the request must be intercepted and the DESIRED_STATE of the resource set accordingly (UP or DOWN). This informs SSM of the request to change the state of the resource (DS=UP or DS=DOWN), and also ensures that the defined DOWN_UP or UP_DOWN action of the resource is always issued. The supplied SSMSTART and SSMSTOP command rules intercept z/OS START and STOP commands for SSM controlled resources.

With these two rules (or rules created with similar logic) enabled, 'S jobname' and 'P jobname' are the only commands ever needed to manually control any active SSM resource.

To START and STOP an SSM controlled resource

1. To start an SSM controlled resource, enter the manual command of 'S jobname'.

   The SSM controlled resource starts, within the SSM DOWN_UP action, the true start command of 'S JOBNAME,PARM=XX' would be correctly issued by SSM if needed.

2. To stop an SSM controlled resource, enter the manual command of 'P DB2MSTR' on an active DB2MSTR region.

   SSMSTOP intercepts the command, sets the DS=DOWN and thus causes SSM to process the specified DOWN_UP action. This action would then invoke the SHUTDB2 OPS/REXX program to initiate the formal DB2 shutdown procedures, all of which was triggered from the manual 'P DB2MSTR' command.

For details on using and implementing this manual intercept method, see the comments within the SSMSTART and SSMTOP rules.

Use a Pseudo CMD Rule to Manually Control SSM Activity

Providing a pseudo command rule similar to the sample SSMCNTL, creates an effective console interface for manually controlling the SSM component.

You can create a controlling command rule to inform SSM to perform tasks that include the following:

- Start and stop resources
- Start/stop specific types of resources
- Recycle resources
- Remove resources from SSM control
- Begin system shutdown

To use a pseudo CMD rule to manually control SSM activity

1. Enable the SSMCNTL rule.
2. Classify a group of resources with a TYPE of TESTCICS within the STCTBL.
3. Issue the pseudo command 'SSM PT TYPE=TESTCICS'.

   SSM is informed to stop all these TESTCICS resources.

For details on using and implementing this manual intercept method (or one with site specific customized functionality), see the comments within the SSMCNTL sample rule.

Invoke STATESET OPS/REXX Program from a Console

You can invoke the supplied STATESET OPS/REXX program through a console command to set the DESIRED_STATE of any SSM controlled resource as well as start and stop dependent prereqs/subreqs of a resource, and changing mode processing values of a resource.

To invoke the STATESET OPS/REXX program

Issue the following command from a console:

*xx*OI STATESET *arguments*

xx

The OSFCHAR OPS/MVS parameter.

arguments

Valid arguments to the STATESET program.

For complete details on using and implementing this manual intercept method, see the comments with the STATESET OPS/REXX program.

Set the Desired State Automatically with a Time Rule

You can write time rules to set the desired state.

Suppose that you want to bring many of your online CICS regions up at a certain time specified in the resource information table. You could add a START_TIME column to the table to identify all CICS regions to be brought up at that time. The following time rule would then start all CICS regions that are supposed to be up at 6:00 a.m.:

```
)TOD 06:00:00
)PROC
   Address 'SQL' "Update STCTBL Set DESIRED_STATE = 'UP'",
     "where TYPE='CICS' and START_TIME='06:00'"
)END
```

Set the Desired State Through the Schedule Manager

SSM also includes a scheduling component that sets the desired state of a resource based on conventional scheduling criteria such as time, date, and day of the week.

The Schedule Manager enables you to define a schedule and periods that set or change the desired state of resources at specific dates and times. For instance, one schedule period might define the desired state for a resource on Monday through Friday, another schedule period might control the desired state on weekends, and a third schedule period might take effect on holidays.

More information:

Using Schedule Manager (see page 315)

Set Desired States Through Checkpointing

SSM resource tables contain a column called CHKPOINT_STATE. The CHKPOINT_STATE value mirrors the desired state, and SSM updates the checkpoint state each time the desired state value changes. Therefore, when SSM becomes active, you can reset the DESIRED_STATE columns to the values they had when SSM was last active.

For example, suppose a system problem occurs and you must perform an emergency IPL. Once SSM becomes active again, it sets the current state and desired state values to table-relative UNKNOWN, and you can then reset the DESIRED_STATE values to what they were before the emergency IPL by copying the values from the CHKPOINT_STATE column into the DESIRED_STATE column.

Set Desired States Through OPSVIEW

You can update the desired state of your resource tables by accessing option 4.11 of the OPSVIEW interface. For more information about this method, see the *OPSVIEW User Guide*.

Set Desired States Through SSMBEGIN

The SSMBEGIN request rule enables you to choose how you want to initialize the DESIRED_STATE column. This gives you complete control over initializing this column each time SSM becomes active and when the SSM mode parameter is upgraded to a more active mode. When executed, this REXX program lists startup options and prompts you to reply. One option allows you to specify the mode (ACTIVE/PASSIVE/INACTIVE) of SSM.

To set the desired states through SSMBEGIN

1. You must enable the SSMBEGIN request rule to be invoked at SSM startup.

   When invoked, the SSMBEGIN request rule displays the following z/OS message:

   ```
   SSMBEGIN start options:
   1) Use current schedule
   2) Use an alternate schedule
   3) Use IPL_STATE as desired state and no schedule
   4) Use previous desired state and no schedule
   5) Use current desired state and no schedule
   ---------- Stateman restarted. Mode=ACTIVE -------------
   Reply with an option number and/or the first letter of a
   valid stateman mode (Active/Passive/Inactive) to replace
   the current mode (i.e. R nn,1/1A/A). Default=1
   015SSMBEGIN ENTER DESIRED STATEMAN STARTUP OPTION (1-5) AND/OR MODE
   A/P/I):
   ```

2. Choose the option that sets the desired states of resources closest to what you want those states to be.

   - If you are restarting the SSM after recycling CA OPS/MVS or in response to an abend, select option 4, because the DESIRED_STATE values in the CHKPOINT_STATE column will most closely reflect what you want the desired states of system resources to be.
   - If you are restarting SSM after it previously shut your system down, determine whether the desired states defined in a schedule or in the IPL_STATE column most closely reflect what you want the desired states to be, and choose option 1, 2, or 3 accordingly.

   Specifying *n*A or *n*P overrides the type of SSM monitoring specified on the CA OPS/MVS STATEMAN parameter. For example, if the STATEMAN parameter is set to ACTIVE and you select option 1P when the SSM starts up, the SSM monitors your system resources passively and does not act when a resource changes state.

More information:

Options for Initializing Desired States (see page 207)

Customize the Startup with SSMBEGUX

The SSMBEGUX REXX program is a user exit called by the SSMBEGIN request rule before the options message is displayed. SSMBEGUX may be customized to automatically select the correct start up options or defaults and bypass the normal options message.

Use this exit for customizing SSM startup rather than modifying the SSMBEGIN request rule. The sample SSMBEGIN program contains programming information for this exit.

Options for Initializing Desired States

You can select the following options to initialize the desired state of SSM resources:

- **Option 1:** Use if you are using the Schedule Manager and all of the resources on your system are defined in the current schedule.

  Causes SSM to reset the desired states of resources to the desired states named in the schedule that is currently active under the Schedule Manager feature. For example, if the current schedule lists the desired state of a DASD as ONLINE, the Schedule Manager changes the desired state of that DASD from UNKNOWN to ONLINE.

- **Option 2:** Use if you are using the Schedule Manager and all of the resources on your system are defined in a schedule different from the active one.

  Causes SSM to reset the desired states of resources to the desired states named in a schedule other than the currently active one. Selecting option 2 produces a message asking you to enter the name of the schedule you want. Once you supply a schedule name, the Schedule Manager uses the desired state values specified in that schedule.

- **Option 3:** Use after a system IPL to reset the desired states of all resources if you are not using the Schedule Manager feature.

  Causes SSM to extract the desired state value for each resource from the value in the IPL_STATE column for that resource.

- **Option 4:** Use to reset the desired states of all resources if you are recycling CA OPS/MVS or restarting it after an abend, and you are not using the Schedule Manager.

  Causes SSM to extract the desired state value for each resource from the value in the CHKPOINT_STATE column for that resource. The values in the CHKPOINT_STATE column are the desired states that resources had before the last CA OPS/MVS or system shutdown.

- **Option 5**

  Use the current desired state value for each resource. Do nothing.

- **Option U**

  Take the default response value specified in the options message.

If you customize SSM for your environment, you also must define your own method for initiating the DESIRED_STATE values for your resources.

Rules to Maintain Current State Values

After you set the values in the DESIRED_STATE columns of your resource tables and SSM dispatches the correct action to get the resource into its desired state, the rules processing (another key element of SSM) takes over.

rules packets

Used by SSM to maintain the current state for resources it manages.

These rules packets intercept messages and update the CURRENT_STATE column using ADDRESS SQL. The rules intercept messages and update the current state of resources as indicated by the message content. A rules packet is a set of rules that work together to perform a common goal. An SSM rules packet might handle all of the messages that pertain to a single resource. For example, the)MSG rule SSMJES2, which is part of SSM, responds to all the $HASP messages that indicate a state change for JES2.

The messages are:

- $HASP085
- $HASP099
- $HASP492
- $HASP493

If the SSMJES2 rule was written as four independent)MSG rules that handle the same messages, then it would still be considered a rules packet.

A rules packet is a logical concept, not a physical entity. When a rules packet is comprised of multiple rules, nothing physically binds the rules together. Rules react to messages that a resource generates and set the CURRENT_STATE value of the resource accordingly when:

- The resource starts
- The resource initializes
- The resource stops or ends
- Error events occur

The OPS.CCLXRULM library on the CA OPS/MVS distribution media contains a number of sample rules packets for started tasks and sample OPS/REXX programs, including the SSMCAAPI rule that processes events from the API interface.

The API interface of CA OPS/MVS lets participating CA products generate common current state APIs. This allows for a single AOF rule, such as the SSMCAAPI rule , to set the CURRENT_STATE of such CA products accordingly within a SSM table. This eliminates enabling and maintaining a unique SSM rule packet to monitor each different CA product. The CASTATE API events are similar to the following:

- State of CA-MIM is UP
- State of CA-MIM is STARTING
- State of CA-MIM is STOPPING

Refer to the documentation of the specific CA product that is being added to SSM to determine if it is generating CASTATE API events to CA OPS/MVS. If it is, then a unique SSM*xxxx* rule packet is not needed to monitor its state within SSM (only need SSMCAAPI rule enabled). This can also be easily determined by ensuring that the CA OPS/MVS APIACTIVE and BROWSEAPI parameters are set to YES prior to the CA product starting, and then viewing the CA OPS/MVS OPSLOG looking for a CASTATE API event (OPSLOG PROFILE for MSGID=CASTATE) being generated by the CA product.

We test each sample rule and OPS/REXX program to ensure that any changes to the state of a resource are represented accurately. However, message identifiers may change, which may affect the operation of sample rules or OPS/REXX programs. If you notice a change in operation, notify CA Customer Support.

If you want to make changes to a sample rule or OPS/REXX program to meet the needs of your site, we strongly recommend that you make a copy of the rule or OPS/REXX program and modify the copy. *Do not make changes to the original sample rule or program.* If you think that your modification would be of interest to other SSM users, contact CA Customer Support.

Your contributions to state rules are encouraged. If you have created an SSM rule or OPS/REXX program that manages the started task of a vendor-supplied product, contact CA Customer Support, who will determine whether your rule or program should be included in a future release of CA OPS/MVS. For assistance, contact Customer Support at http://ca.com/support.

The sample code that follows shows you a sample rules packet:

```
)MSG DFH*
)PROC
/*********************************************************************/
/*                                                                   */
/*            PROPRIETARY AND CONFIDENTIAL INFORMATION               */
/*                AND INTELLECTUAL PROPERTY OF CA.                   */
/*            Copyright (C) 2009 CA. ALL RIGHTS RESERVED.            */
/*                                                                   */
/*   NAME       - SSMCICS                                           */
/*   PURPOSE    - Set the current state of CICS regions based on     */
/*                message traffic.                                   */
/*   RELATED    - STATEMAN                                          */
/*   GLOBALS    - None                                              */
/*   PARMS      - None                                              */
/*   KEYWORDS   - None                                              */
/*   LANGUAGE   - OPS/REXX                                          */
/*   HISTORY    - 27 APR 2009 GEM  - Original implementation        */
/*                                                                   */
/*   NOTES      -                                                   */
/*                                                                   */
/*********************************************************************/
If Opsinfo('EXITTYPE') ^== 'MVS' Then Return
job   = msg.jobname
msgid = msg.id
Select
/*-------------------------------------------------------------------*/
/* DFHSI1500 applid element startup is in progress for CICS ...      */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----*/
  When msgid == 'DFHSI1500' then
    If wordpos('startup is in progress',msg.text) > 0 then
      Address SQL "Update STCTBL set current_state='STARTING'",
                    "where jobname=:job and type='CICS'"
/*-------------------------------------------------------------------*/
/* DFHSI1517 applid Control is being given to CICS                   */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----*/
```

```
  When msgid == 'DFHSI1517' then
      If wordpos('Control is being given to CICS',msg.text) > 0 then
        Address SQL "Update STCTBL set current_state='UP'",
                    "where jobname=:job and type='CICS'"
/*--------------------------------------------------------------------*/
/* DFHTM1715 applid product is being quiesced by userid ...           */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----*/
  When msgid == 'DFHTM1715' then
    Address SQL "Update STCTBL set current_state='STOPPING'",
                "where jobname=:job and type='CICS'"
/*--------------------------------------------------------------------*/
/* DFHKE1799 applid TERMINATION OF CICS IS COMPLETE                   */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----*/
  When msgid == 'DFHKE1799' then
    Address SQL "Update STCTBL set current_state='DOWN'",
                "where jobname=:job and type='CICS'",
                "and current_state <> 'ERROR'"
/*--------------------------------------------------------------------*/
/* Not interested in this message                                     */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----*/
  Otherwise nop
End
Return
```

Understanding Transient Resource States

The state of the resource goes from STARTING to UP without SSM having to execute any STARTING_UP action, even if the actions table defines a STARTING_UP action.

transient states

States such as STARTING, which change to something else without SSM intervening.

stable states

States that do not change unless SSM takes action.

The events that cause a resource to change state usually are external events that rules can interpret, such as system messages or commands the operator issues. However, actions defined in the action table of the SSM also can trigger state changes.

A single SSM action can also cause two or three events to occur in sequence. For example, when the TSO resource is DOWN and the desired state is set to UP, SSM executes a START TSO action. This action causes two separate state-changing events to happen for the same resource:

- The message $HASP373 TSO STARTED triggers a rule that sets the current state to STARTING.
- The message IKT007I triggers a rule that sets the current state to UP.

ops--How Transient States Work

States such as STARTING, which change to something else without SSM intervening, are called *transient states*. States that do not change unless SSM takes action are called *stable states*.

SSM reacts to changes in the finite state machine by executing actions defined in its actions table. So, the DOWN_UP transition must initiate an action to make the resource leave the (stable) DOWN state.

Any action defined for a *transient* state may or may not execute, depending on the timing of events, nor does it need to execute for things to happen:

- The action defined for a transient state may not execute if the triggering event occurs before SSM discovers that state. If SSM does queue and execute the action, the current state of the machine may have changed by the time the action routine begins. Therefore, if you define any actions involving transient states such as STARTING, make them actions that do not necessarily have to execute.

  For example, you may want to define an action for a transient state such as STARTING that will execute an automation procedure that checks in two minutes to see if the current resource state is UP and issues an error message if the state is anything else. This is a safe, prudent action to take, because if the rule executes, an error condition may exist. Actions such as issuing a VARY command or setting a critical global variable, or any action that is *required* for the resource that is starting, are *not* safe to take for the STARTING transition, because these actions may not execute (and if they do, they may execute after the state of the resource has already changed to UP).

- Actions defined for transient states execute mostly for resources that are defined in the first table listed in the directory table. Remember that the directory table lists the resource tables in alphabetical order. But, the lower resources are in the table list order in the directory table, the more likely it is that external state changes will occur before the actions defined for transient states execute, or that these actions never execute.

- If you have critical actions that must be taken for transient resource states, the safest place to specify those actions is in the rules packet for the applicable resource type. In our example, if you want to take a critical action at START TSO time, execute this action at the same point where you detect and set the STARTING state. Do this before updating the table row for the appropriate resource.

How SSM Decides What Action to Take

Various events can cause the current and desired operating states of a resource to differ. Therefore, the values in the ACTION_CURRENT and ACTION_DESIRED columns of your action table should take different types of resource state changes into account. For example, you probably want to specify the following pairs of values in the ACTION_CURRENT and ACTION_DESIRED columns of your action table:

ACTION_CURRENT Column Value	ACTION_DESIRED Column Value	Meaning of This Value Pair
UNKNOWN	null	The current state of a resource is UNKNOWN and its desired state is anything else.
DOWN	UP	The current state of a resource is DOWN and its desired state is UP.
UP	DOWN	The current state of a resource is UP and its desired state is DOWN.
ERROR	UP	The current state of a resource is ERROR and its desired state is UP.
ERROR	DOWN	The current state of a resource is ERROR and its desired state is DOWN.
STOPPING	UP	A resource is stopping and its desired state is UP.
STARTING	DOWN	A resource is starting and its desired state is DOWN.

How to Specify and Store Actions

Actions tell the SSM engine what steps need to be taken when:

- A resource state changes
- An action event occurs
- A global event occurs

To specify actions, use the following guidelines:

- Use the table editor to insert action clauses into the ACTION_TEXT field. The field is type VARCHAR and can accept a maximum of 450 characters.
- Combine multiple clauses into a single ACTION_TEXT field by using a semicolon as a separator. If it is necessary to include a semicolon in a clause, enter two consecutive semicolons.

To store actions, use the following guidelines:

- Actions are always stored in the ACTION_TEXT field of an action table.
- Store actions for global events in the global action table. There is only one global action table, and it is referenced only during global events.
- Store actions for both the resource state changes and for process events in a resource action table. There can be multiple resource action tables.

Action Clauses

There are eight possible action clauses, which are listed below.

Following are descriptions of the eight action clauses:

CLIST("*cmdname prm1,prm2,...*")

Uses %CMDNAME to send the specified CLIST command and optional arguments to the CLIST interpreter. The command runs on a server task level, so the CLIST script is permitted to execute commands that run for a long period. The CLIST script must exist in either the SYSPROC or SYSEXEC concatenation.

Note: The CLIST clause is included for compatibility. For new code, use TSOCMD to invoke OPS/REXX programs.

Example: CLIST("STARTDB2") runs a CLIST named STARTDB2.

EVRULE("*rulename prm1,prm2,…*")

Dispatches an AOF request rule and waits for the rule to complete. EVRULE is the same as RULE, except that the return code from the request rule is checked. If the return code is 500 or higher, then any clauses following the EVRULE are not executed. The return code that is checked is set on the return statement in the request rule.

WARNING! The rule triggered by this action executes to its conclusion before control returns to the action. To avoid slow response time from the SSM engine, avoid logic that could create lengthy wait times.

Example: EVRULE("SSMRECYCLE &NAME");MVSCMD("START &JOBNAME") dispatches the SSMDOWN rule and check the return code from the rule. If the return code is equal to or greater than 500, then action processing is terminated without executing the MVSCMD clause.

MVSCMD("*mvs command text*")

Issues a z/OS command and does not wait for output. The name of the console that will be used to execute the command is specified in the OCCONSOLENAME parameter.

Example: MVSCMD("START &JOBNAME,SUB=MSTR") issues a start command and does not return results.

PROCESS("*event1,event2,…*")

Used only in a process event action. Usually, it should be used in the SELECT process event action, but there could be valid uses for a process clause in other global events actions. For example, it could be used to suppress an event that was enabled earlier in the processing of the resource state change event.

Process tells the SSM engine which process events to dispatch as SSM processes a resource state change. For an explanation of process events, see Process Events in this chapter.

Example: PROCESS("MATCH,MPREREQ") enables the MATCH and MPREREQ events for the current resource.

RULE("*rulename prm1,prm2,…*")

Dispatches an AOF request rule and waits for the rule to complete.

WARNING! The rule triggered by this action executes to its conclusion before control returns to the action. To avoid slow response time from the SSM engine, avoid logic that could create lengthy wait times.

Example: RULE("SSMDOWN &NAME") dispatches SSMDOWN request rule.

SETCOL("*column name*, *column value*")

Updates the indicated column of the current resource with the specified value. You can use SETCOL to store a column value directly in an action. The column value operand is limited to 140 characters, and must be enclosed in single quotes if it contains blanks or other special characters such as quotes, parenthesis, commas, or equal signs. To set a column value to null, specify a value of NULL for the column value operand.

Note: If other action clauses follow SETCOL in the same action text, then references to the updated column will see the original value of the column, not the updated value. The new value will not be visible until the resource is reselected. This is consistent with the capability of other SSM action clauses.

Example: SETCOL("DESIRED_STATE,&DOWN_STATE") sets the desired state of the current resource to the table-relative DOWN state.

TSLCMD("*cmdname prm1*,*prm2*,…")

The command and parameters specified are routed to a CA OPS/MVS TSL Server address space. This is an asynchronous action, and therefore will *not* degrade SSM response time. This action is primarily used to invoke OPS/REXX programs in which long running logic is needed to manage a SSM Controlled resource.

Example: TSLCMD("OI PROGRAM(MOVECICS) ARG(&JOBNAME)") runs an OI TSO command, which invokes an OPS/REXX program named MOVECICS. The program MOVECICS must be stored in the SYSEXEC concatenation, or as a compiled OPS/REXX program in the OPSEXEC concatenation within the OPSOSF server Procedure.

For more information on defining and controlling CA OPS/MVS TSL server address spaces, see the *Administration Guide*.

TSOCMD("***cmdname prm1***,***prm2***,…")

The command and parameters specified are routed to a CA OPS/MVS TSO Server address space. This is an asynchronous action, and therefore will *not* degrade SSM response time. This action invokes OPS/REXX programs in which in depth logic is needed to start or stop the SSM controlled resource.

Example: TSOCMD("OI PROGRAM(SHUTCICS) ARG(&JOBNAME)") runs an OI TSO command, which invokes an OPS/REXX program named SHUTCICS. The program SHUTCICS must be stored in the SYSEXEC concatenation, or as a compiled OPS/REXX program in the OPSEXEC concatenation within the OPSOSF server Procedure.

For more information on defining and controlling CA OPS/MVS TSO server address spaces, see the *Administrations Guide*.

TSPCMD("cmdname prm1,prm2,…")

The command and parameters specified are routed to a CA OPS/MVS TSP Server address space. This is an asynchronous action, and therefore will *not* degrade SSM response time. This action is primarily used to invoke OPS/REXX programs by SSM that must have execution priority over other OPSOSF triggered automation.

Example: TSPCMD("OI PROGRAM(PRODABND) ARG(&JOBNAME)") runs an OI TSO command, which invokes an OPS/REXX program named PRODABND. The program PRODABND must be stored in the SYSEXEC concatenation, or as a compiled OPS/REXX program in the OPSEXEC concatenation within the OPSOSF server Procedure.

For complete details on defining and controlling CA OPS/MVS TSP server address spaces, see the *Administrations Guide*.

Complex Actions

A series of action clauses may be sufficient to do all the processing you need in an action, but you can use TSOCMD to dispatch an OPS/REXX program of arbitrary complexity. Since the program will run in a server subtask, the SSM engine is free to proceed with another action before the program completes. CLIST and REXX action clauses also dispatch the called program on a server subtask. RULE and EVRULE run an AOF request rule. The rule will run to conclusion before your action regains control, so long-running rules should be avoided.

Specify Variables in Action Clauses

CA OPS/MVS enables you to specify variables in an action clause. This product enables variables by assigning values to the variables when SSM initiates the action in the clause.

You can specify action text variables in two ways. The first method is compatible with previous versions of SSM, and the second method follows the rules of REXX host command string processing.

When using the first method (ampersand prefixed variable name), CA OPS/MVS provides a substring functionality to extract desired substring from the variable name.

To specify variables in action clauses.

- The first method is compatible with previous versions of SSM, you specify variables in a manner similar to TSO/E CLIST syntax. Follow this procedure:
  - Place an ampersand (&) as the first character of the variable name.
  - The first invalid character terminates the variable name.
  - A period (.) following the variable name indicates concatenation with subsequent text.
  - To include an ampersand character (&) as part of the text, enter two ampersands together.
  - To use a substring functionality, specify starting position and length in the following notation: &varname(start,length).
  - If the variable name is not a column name in the directory or resource table, the variable name is assumed to be a CA Automate rules variable that has been processed by the CA OPS/MVS rules converter. The prefix table established by the ATM*SCOPE*nn* parameters are searched and the appropriate converted variable name is used to find the variable value. If the variable does not exist, a null string is substituted. Quotes have no special meaning as the command text is being scanned for variables.
- The second method for specifying action text variables follows the rules of REXX host command string processing. Follow this procedure:
  - All text in quotation marks, either single (') or double ("), remain as entered.
  - Text outside of quotes can be treated as a variable name.
  - Strings of valid variable names are evaluated for substitution as column names or global variables.
  - Characters that are not valid variable names remain as entered.
  - Multiple blanks are reduced to a single blank.
  - A uninitialized variable is treated as the value of its own name.

When CA OPS/MVS evaluates action text, it first evaluates according to the TSO/E CLIST method, and then evaluates again using the REXX syntax method. To prevent an inadvertent variable substitution, enclose all command text in quotes in the action keyword specification.

Example: Variables that are specified in action clauses.

The following command text string is enclosed in quotes to prevent variable substitution.

```
TSOCMD("OPSCMD COMMAND(START &JOBNAME..&JOBNAME) NOOUTPUT
SUBSYS("GLOBAL0.MYSUB")")
```

In this example, if the JOBNAME column in the resource table has the value of RMF and the GLOBAL0.MYSUB variable has the value OPSX, then the following command is submitted to the server:

```
OPSCMD COMMAND(START RMF.RMF) NOOUTPUT SUBSYS(OPSX)
```

For a DB2 subsystem that uses a quotation mark (") as a command character, suppose that you have this action text:

```
TSOCMD("OPSCMD COMMAND(OI STOPDB2 CMDCHAR("")) JOBNAME("JOBNAME")")
```

This text yields the following command:

```
OPSCMD COMMAND(OI STOPDB2 CMDCHAR(") JOBNAME(DSN2MSTR)
```

The following command text string is the DB2 START command.

```
MVSCMD("-&JOBNAME(1,4) START DB2”)
```

To start a DB2, use the DB2 command prefix which in most cases is first four letters of the master address space. In this example, if the JOBNAME column in the resource table has the value of DB2PMSTR following command is submitted to MVS:

```
-DB2P START DB2
```

Shorthand global variables can also be used. These variables have the format &.varname. SSM resolves these variables by replacing the ampersand (&) with a system-specific prefix, and then treating them as global variables. The system prefix is set by the SSMGLVPREFIX parameter, which you set independently on each system.

During the REXX substitution phase of action text processing, the & can be omitted if the variable name is not embedded in a quoted string.

Built-in Variables

Built-in variables that begin with the prefix SSM! are designed to provide global and process exits with certain data that is not found in RDF columns or global variables.

The following describes the built-in variables:

SSM!XRESNAME

A short form of the concatenation of the variables &SSM!PRISYS..&SSM!TABLE..&SSM!NAME which is usually passed as an action parameter for the XPREREQ and XSUBREQ process events.

SSM!USERDATA

Contains the job name and program name of the SQL statement issuer that caused the global event.

SSM!TABLE

Contains the name of the resource table or SSM directory table for a global event or process event or the current resource table being processed by SSM for process events and normal resource actions.

SSM!RESNAME

A short form of the concatenation of the variables &SSM!TABLE..&SSM!NAME which is often passed as an action parameter.

SSM!PROCESS

Contains the name of the global or process event or ACTION for normal resource action processing.

SSM!PRISYS

Contains the primary or local system name of the resource in the format *system.subsys*. If a column called PRIMARY_SYSTEM does not exist in the resource table, then the local system name is always returned.

SSM!PREREQ

Contains the text of the prerequisite to be evaluated by the XPREREQ process event.

SSM!NAME

Contains the name of the resource or managed table name for a global event or process event or the current resource name being processed by SSM for process events and normal resource actions.

SSM!MAXRC

Contains the maximum return code that has occurred for any global event or resource action table processing when multiple actions are defined in the text column of the action table.

SSM!LASTRC

Contains the return code from the last completed action.

SSM!IPL

Contains YES or NO. YES implies the first call to SSMBEGIN and the BEGIN global event since IPL. The value is NO at all other times.

SSM!COLUMN

Contains the name of the first monitored column that was updated by the SQL statement that triggered an UPDATE global event.

Notes:

- The only valid variables for the BEGIN global event are SSM!IPL, SSM!MAXRC, and SSM!LASTRC.
- Resource and directory table column variables and SSM!PRISYS are not available for action text substitution in global event actions. The REXX program invoked for a global event action must read the resource or directory table to obtain any column data.
- The SSM!COLUMN variable is only valid for the UPDATE global event.
- The SSM!PREREQ variable is only valid for the XPREREQ process event.
- The SSM!USERDATA variable is only valid for all global events except BEGIN.
- The SSM!MAXRC and SSM!LASTRC values are always valid and initialized to zero before any action processing begins.

Substituting Data

Action text variables consist of an ampersand (&) followed by a 1- to 32-character name.

CA OPS/MVS substitutes data into the variable in these ways:

- From the current row of the directory table.
- From the current row of the resource information table.
- From a global variable, as follows:
  - If the variable is the name of a column in the directory table, then the value of that column for the current row becomes the value for the variable.
  - If the variable has the name &SSMTABLE, its value is the current name of the resource information table.
  - If the variable is the name of a column in the current resource information table, then the value of that column for the current row becomes the value for the variable.
  - If the variable is a global variable, the value of the global variable becomes the value of the action clause variable.
  - If the variable is a system symbol, the value of the symbol becomes the value of the action clause variable.
  - In all other cases, CA OPS/MVS substitutes a null string for the variable when you use the CLIST method, and the variable name for the REXX method.
- When using substring notation - &varname(start,length) as follows

  start

  This parameter is the starting position parameter, which is determined from the substring beginning position. A value of 1 indicates the first value of the string. When the position exceeds the length of the original string, then only pad characters are returned.

  ***length* (Optional)**

  This parameter determines the length of substring to extract. When this argument is absent, the remainder of the string is returned. When the length exceeds the number of characters remaining, pad characters are added at the end.

  Note: Pad character is a blank.

Empty-string and NULL Actions

If the action specified in the ACTION_TEXT column is an empty string, the SSM interprets it as a valid action (in this case, a do nothing action). No other processing occurs.

If the ACTION_TEXT column contains a NULL value, the SSM interprets it as a non-action and continues searching the action table for another match.

ACTMODE Column

The ACTMODE column in a resource table provides a way for the automation designer to specify different actions for the same mismatch event based on any relevant criteria. For example, a DOWN/UP mismatch might be responded to differently immediately following an IPL when some subsystems are not ready, than it would after the system has reached full production.

The ACTMODE column strongly influences the action selection process when SSM responds to a resource state change.

When the value of the ACTMODE column of a resource is changed, the SSM engine immediately selects that resource for processing, even if there is no state mismatch. This selection occurs regardless of what caused the value of the ACTMODE column to change. For example, the following actions all cause selection to occur:

- An SQL UPDATE statement in an AOF rule
- A manual change through the table editor
- Use of the SETCOL action clause in action text

The purpose of the ACTMODE column is to define special actions. For example, consider the relationship between CICS and IMS, where IMS is a prerequisite of CICS and CICS is a subrequisite of IMS. Suppose you want to restart IMS without shutting down CICS. To do this, you could cause the IMS resource to be selected for processing by SSM by setting ACTMODE='BOUNCE' for IMS, and then supplying the following action to your action table:

```
 ACTION_PROCESS ACTION_CURRENT ACTION_DESIRED ACTION_MODE ACTION_RES_TYPE ACTION_TEXT
1 ACTION         UP             UP             BOUNCE                      RULE("BOUNCE &JOBNAME")
```

The rule BOUNCE ignores the subrequisite relationship between CICS and IMS and cycles IMS as requested.

More information:

How SSM Decides What Action to Take (see page 213)

Search Order for Action Tables

Two kinds of events use the resource action table:

- State change events

  State change events take action in response to resource state changes.

- Process events

  Process events affect what action the SSM engine takes for the state change event that it is currently processing.

The SSM engine applies different rules when evaluating these two kinds of events.

Evaluating Resource State Change Events

For an action to be taken in response to a resource state change event, the following values must be set:

- The value of the ACTION_PROCESS column of the action table must be set to ACTION
- The value of the ACTION_CURRENT column of the action table must match the current state of the resource.
- Any column in an action table that is set to a non-null, non-blank value must match the value of the corresponding column in the resource table.

For example, if the value of the ACTION_DESIRED column of an action table is READY and the desired state of a resource is RUNNING, then the action is not eligible for execution. However, if the desired state of the resource equals the value of the ACTION_DESIRED column, then the action is eligible for execution and its score is incremented by the weight assigned to the ACTION_DESIRED column, which is 16. If the value of the ACTION_DESIRED column is null or blank, then the action is eligible for execution regardless of the desired state of the resource; however, no increment is added to the score. The same evaluation is carried out for all four weighted columns listed in the table on the following page.

Note: Actions are selected based on their score. The action with the highest score is the one executed in response to the resource state change event.

The least specific action of all possible actions sets the value of all four columns listed below to null or blank. The score assigned to this type of action is zero. The most specific action of all possible actions sets the value of all four columns to a non-null, non-blank value. The score assigned to this type of action is 29 or 30, depending on whether the value of the NAME column or the value of the TYPE column matched the value of the ACTION_RES_TYPE column. Only one of these columns can match the ACTION_RES_TYPE column, not both.

Following is a list of the action table columns for state change events and their corresponding weights:

Column	Weight
ACTION_DESIRED	16
ACTION_MODE	8
ACTION_RES_TABLE	4
ACTION_RES_TYPE	1 or 2

As mentioned above, the ACTION_RES_TYPE column is evaluated once against the NAME column and once against the TYPE column to see if a match occurs. If the value of the NAME column matches the value of the ACTION_RES_TYPE column, then a value of 2 is added to the action score. The TYPE column is an optional column in the resource definition. If it is present and if its value matches that of the ACTION_RES_TYPE column, then a value of 1 is added to the action score.

Evaluating Process Events

For an action to be selected in response to a process event, the value of the ACTION_PROCESS column in the action table must match the process event type. The eligible action event names are SELECT, MATCH, PREREQ, SUBREQ, MPREREQ, MSUBREQ, XPREREQ, and XSUBREQ. If the value of the ACTION_PROCESS column matches one of these reserved names for one or more actions, then the action assigned the highest score is executed. The scoring of process events is similar to that of state change events except that the action tables of process events contain an ACTION_CURRENT column.

Following is a list of the action table columns for process events and their corresponding weights:

Column	Weight
ACTION_CURRENT	32
ACTION_DESIRED	16
ACTION_MODE	8
ACTION_RES_TABLE	4
ACTION_RES_TYPE	1 or 2

Example of Action Selection

This example shows the state mismatch action selection. Some columns were omitted for clarity.

```
NAME       CURRENT_STATE DESIRED_STATE ACTMODE TYPE
  RES1     DOWN          UP                    STC
  RES2     DOWN          UP                    STC
  RES3     DOWN          UP            BOUNCE  STC
  RES4     STARTING      UP                    STC
```

Following is an example of the actions that were selected in response to the above state mismatch:

```
  ACTION_PROCESS ACTION_CURRENT ACTION_DESIRED ACTION_MODE ACTION_RES_TYPE ACTION_TEXT
1 ACTION         DOWN           UP                                        MVSCMD("START &JOBNAME")
2 ACTION         DOWN           UP                        RES2            MVSCMD("S DJOB,PARM='IPL'")
3 ACTION         DOWN                          BOUNCE                    TSOCMD("OI BOUNCE")
4 ACTION         DOWN           UP             BOUNCE                    TSOCMD("OI REBOUNCE")
5 ACTION         STARTING       UP                                        SETCOL("CURRENT_STATE,DOWN")
```

In the previous example, the following actions are taken in response to the resource state change events:

- For RES1-Action 1 is selected with a score of 16. Action 5 is not considered for this resource because the current state does not match the ACTION_CURRENT column of the action table. Actions 2, 3, and 4 are bypassed because there are non-null columns in the action table that do not match the resource.
- For RES2-Action 2 is selected with a score of 18. Action 1 also matches the resource; however, its score is 16. Actions 3 and 4 are rejected because the ACTION_MODE column is non-null and it does not match the resource. Action 5 is not even considered because the current state does not match the value of the ACTION_CURRENT column of the action table.
- For RES3-Action 4 is selected with a score of 24. Actions 1 and 3 are also matches with scores of 16 and 8 respectively.
- For RES4-Action 5 is selected with a score of 16. No other actions are considered.

Process Events

The process event facility allows users to influence resource processing when state transitions occur. This facility allows users to enhance the decision making process in the SSM engine. For example, a process event may execute a user-supplied action that determines whether a prerequisite resource is available on a remote system. There are eight events at critical points in SSM where an event can occur.

The PROCESS action clause gives users control over process action invocation on a per event and per resource basis. The PROCESS action command is discussed in the next section.

Following are the process events and their descriptions. SSM!PROCESS will be set to the event name when the action gets control.

MATCH

Occurs once when the current state equals the desired state for the first time. Typically, a MATCH action can be used to start a dependent resource. In this case, dependent refers to a resource that lists the current resource as a subrequisite. For example, if the desired state of VTAM is UP, then the MATCH exit might start CICS.

SELECT

Occurs for each resource every time the SSM engine finds a reason to process the state of a resource. SELECT occurs if the current state does not match the desired state, if the missing prerequisite column or previous state column is not null, or when the ACTMODE column has changed for the resource. A SELECT action should be used to issue a PROCESS command that enables or disables subsequent process events as needed to implement your automation strategy.

MPREREQ

Occurs once if any missing prerequisites remain after usual prerequisite processing. This event allows a user-supplied action to apply its own criteria to the missing prerequisites. The event action must be a synchronous action (EVRULE or RULE) that updates the MISSING_PREREQ column to remove a prerequisite that is UP or possibly adds a prerequisite that is DOWN.

Notes:

- XPREREQ events execute before the MPREREQ event.
- If the status of a MINOF statement is DOWN, then SSM stores the entire text of the MINOF statement in the missing prerequisite column. In the case of a MINOF statement, the event action must evaluate the entire MINOF statement again to determine its true state.

XPREREQ

Occurs once for every prerequisite name that includes a system name such as SYST01.OPSW.TABLE.NAME. If the action determines that a prerequisite is missing, then the action must return to SSM with a non-zero return code. This causes SSM to add the prerequisite to the missing prerequisite list.

Note: The action for this event should be a synchronous action (EVRULE or RULE) so the user code can return a non-zero return code if the prerequisite is missing. If you execute an asynchronous action such as REXX, then the return code passed back to SSM is zero and the prerequisite is not added to the missing prerequisite list, even if your asynchronous REXX code determines that the prerequisite is missing.

MSUBREQ

Occurs once for any number of missing subrequisites if any missing subrequisites are found by SSM standard processing. An MSUBREQ action must be an asynchronous action (EVRULE or RULE) that can use its own logic to determine subrequisite status and may override the SSM subrequisite decision by updating the MISSING_PREREQ column.

Note: XUSBREQ events execute before the MSUBREQ event.

XSUBREQ

Occurs once for the resource being evaluated after SSM processing, even if multiple subrequisites are missing. Action code must be a synchronous action (EVRULE or RULE) that must discover subrequisite dependence on other systems, and update the MISSING_PREREQ column for the current resource if any missing subrequisite is found, or if a missing subrequisite is satisfied.

The PROCESS Action Clause

As mentioned in the previous section, the PROCESS action command gives users control over the process exit invocation on per decision point and per resource basis. The PROCESS action command is issued in the SELECT exit. Its purpose is to selectively enable each of the process exits for the current resource by providing a list of process names. Following is an example of the format of the command:

```
PROCESS(MATCH,XPREREQ,XSUBREQ)
```

The following are valid names that can be specified on the PROCESS command:

DEBUG

A miscellaneous value. DEBUG enables the same diagnostic messages as the SSMDEBUG parameter, but for the current resource only.

MATCH

A positive name that enables the action to be invoked.

MPREREQ

A positive name that enables the action to be invoked.

MSUBREQ

A positive name that enables the action to be invoked.

NOACTION

A negative name that prevents the SSM engine from asserting any action for the resource event.

NOPREREQ

A negative name that prevents the SSM prerequisite processes and their affiliated actions from being performed.

NOSUBREQ

A negative name that prevents the SSM subrequisite processes and their affiliated actions from being performed.

SQLTRACE

A miscellaneous value. SQLTRACE produces the same diagnostic messages as the STATETRACE parameter for the current resource.

XPREREQ

A positive name that enables the action to be invoked.

XSUBREQ

A positive name that enables the action to be invoked.

SSM Action Processes Default Logic

The SSM action table, created using the Snapshot facility (default name of STCTBL_ACT), provides the following default actions:

- Handles failures that may occur with SSM controlled STCs
- Provides a technique to control and monitor STCs that start, perform some type of work, and then end (does not remain active)

This section describes the default logic supplied within these actions. Review and customize the default actions to fit the requirements of your SSM environment or remove the triggering actions as documented from the action table, if they are not desirable.

How SSMRETRY Limits a Repetitive Action Sequence

The SSMRETRY request rule is a technique used to limit a repetitive action sequence from occurring, such as starting a problem resource.

The SSMRETRY request rule performs the following actions and causes these effects:

- Parameters passed to SSMRETRY specify how many start attempts should be made within a defined time period.

  For example, if a resource is started on a system by SSM as a result of DESIRED_STATE='UP', the resource starts on the system (CURRENT_STATE='STARTING') but then terminates abnormally (CURRENT_STATE='DOWN'), the SSMRETRY logic limits SSM from attempting another start of the resource and sets the current state to failed (CURRENT_STATE='FAILED').

- Parameters passed to SSMRETRY specify a maximum time that a resource should reach its initialization state. If a resource does not reach its initialization state (CURRENT_STATE='UP' as set by associating SSM rule packet), then the SSMRETRY logic will set the current state to timeout (CURRENT_STATE='TIMEOUT').

  For example, if the maximum initialization time for JES2 is set to five minutes, when SSM starts JES2, and if it goes into some internal loop (never produces $HASP492), then after five minutes the logic created by SSMRETRY will set the CURRENT_STATE = 'TIMEOUT' for JES2.

- The SSMRETRY request rule is invoked as a nested action within the DOWN_UP action process for a resource such as:

  ```
  EVRULE("SSMRETRY &SSMTABLE &JOBNAME,1,300,START UP");MVSCMD("START &JOBNAME")
  ```

This START UP action process is limited to one start attempt and the resource should reach its initialization state (CURRENT_STATE = UP) within 300 seconds. These parameters can be change accordingly. The invocation of the SSMRETRY as an EVRULE action allows the code to inform SSM to either process or bypass the nested or subsequent start resource action. (start resource)

For further details and complete implementation guidelines, see the comments within the SSMRETRY request rule.

How SSMFAIL Responds to SSM Resource Start-up, Termination, and Timeout Conditions

The SSMFAIL request rule generates an alert to start-up failures as determined from logic within the SSMRETRY routine (CURRENT_STATE=FAILED, or CURRENT_STATE=TIMEOUT). The SSMFAIL request rule also generates an alert when the logic with the SSMEOM rule sets CURRENT_STATE=TERM upon an initialized SSM resource abnormally terminating.

- For FAILED conditions (SSMRETRY action retry limits exceeded), the SSMFAIL request rule is triggered through the FAILED_UP action text specification:

```
ACTION_CURRENT   ACTION_DESIRED   ACTION_TEXT
FAILED           UP               RULE("SSMFAIL &NAME FAILED &SSMTABLE")
```

  The default SSMFAIL logic for this condition leaves the resource in the FAILED_UP condition and produces a highlighted message similar to the following:

```
OPSNOTIFY DB2MSTR failed and will not be restarted. Action retry limits
defined in SSMRETRY have been exceeded.
```

- For TIMEOUT conditions (SSMRETRY detected a resource has started but it has not reached its 'UP' state), the SSMFAIL request rule is triggered through the TIMEOUT_UP action text specification:

```
ACTION_CURRENT   ACTION_DESIRED   ACTION_TEXT
TIMEOUT          UP               RULE("SSMFAIL &NAME TIMEOUT &SSMTABLE")
```

  The default SSMFAIL logic for this TIMEOUT_UP condition produces the highlighted message similar to the following:

```
OPSNOTIFY DB2MSTR has not reached initialization as defined within SSMRETRY
Parameters.
```

- For TERM conditions (SSMEOM detected an initialized SSM resource that was in the states CURRENT_STATE='UP' and DESIRED_STATE='UP' has abnormally terminated), the SSMFAIL request rule is triggered through the TERM_UP action text specification:

```
ACTION_CURRENT   ACTION_DESIRED   ACTION_TEXT
TERM             UP               RULE("SSMFAIL &NAME TERM &SSMTABLE")
```

The default SSMFAIL logic for this TERM_UP condition produces the highlighted message similar to the following:

```
OPSNOTIFY DB2MSTR has abnormally terminated. SSM is restarting.
```

For further details and complete implementation guidelines, refer to the comments within the SSMFAIL request rule.

How SSMHOOK Controls Quick Work Cycles

Various products are started at IPL time that simply start, perform some work, and then quickly end. These jobs are classified as hook jobs within SSM and need to be treated differently than those SSM resources that start, initialize, and remain active until they are requested to shutdown. The SSMHOOK request rule can be used to manage these hook jobs within SSM.

The SSM action tables created from the Snapshot facility in release r11.6 or higher of CA OPS/MVS will insert the following default actions for a HOOKSTC resource:

- UP_DOWN
- DOWN_UP
- HOOKED_UP

These actions can be repeated and modified accordingly, to manage any hook job.

Refer to the comments within the SSMHOOK request rule for further details and complete implementation guidelines for handling hook jobs within SSM.

SSM Global Events

There are two types of SSM global events, RDF monitor events (ADD, DELETE, and UPDATE) and the BEGIN event. Users can take action for these events by adding entries to the global action table.

RDF Monitor Events

SSM monitors the resource directory table and all tables named in it for changes. If a column has the monitored attribute, then SSM monitors that individual column as well. When a monitored column is changed or a row is added or deleted from a managed table, the RDF stores information about the change and notifies SSM through a global event. As a result of the global event being issued, SSM comes out of its usual wait state and processes the information stored by the RDF. A user-defined action can now run when these events occur.

RDF-triggered events occur after the table has been modified, not synchronously with the change. The corresponding action can react to changes; however, it cannot prevent them.

Note: Auxiliary tables are not considered to be SSM managed tables, and changes to them do not create global events.

BEGIN Event

SSM initialization occurs when SSM is started, or when the status of SSM is changed from INACTIVE to ACTIVE or PASSIVE, or from PASSIVE to ACTIVE. During initialization, SSM attempts to execute a global action named BEGIN. This global action is optional. Its purpose is to obtain the status of resources that exist on an external processor and are potential prerequisites for local resources; however, other initialization can be performed in this action at the discretion of the user. You will not be able to install and run this application until you stop using your action, or until you combine the CA-supplied action with your own.

How the BEGIN Event Works

Only one BEGIN action may be specified in the SSMV2_GBLEXIT_TBL global action table. The BEGIN event runs after the SSMBEGIN request rule.

In the action, SSM!PROCESS is always set to BEGIN and SSM!IPL is always set to YES for the first pass following a system IPL. Otherwise, SSM!IPL is NO.

Example: BEGIN Event

Following is an example of a BEGIN event that will execute a request rule called SSMSTART, which is passed to the SSM!IPL built-in variable:

```
SSMV2_GBLEXIT_TBL ------------ TABLE DATA EDITOR ----------- COLUMNS 00001 00072
Command ===>                                                  Scroll ===> CSR
COL--> EXIT_EVENT EXIT_RES_TABLE EXIT_RES_NAME  EXIT_ACTION
*********************************TOP OF DATA*********************************
000001 ADD
000002 BEGIN                                    RULE("SSMSTART &SSM!IPL")
000003 DELETE
000004 UPDATE
*******************************BOTTOM OF DATA********************************
```

Using SSM Global Events

The SSMGLOBALEXITS parameter controls whether the global action is enabled. This parameter defaults to NO, the global action is not enabled.

You must also create a global action RDF table, which contains the following four required columns:

EXIT_EVENT	CHAR(8)	PK UC	NULL
EXIT_RES_TABLE	CHAR(18)	PK UC	
EXIT_RES_NAME	CHAR(18)	PK UC	
EXIT_ACTION	VARCHAR(450)		NULL

Additional columns are allowed.

If the global action is enabled by setting the SSMGLOBALEXITS parameter to YES and the global action RDF table does not exist, then the SSM global action process does the following:

- Creates an empty table containing the four required columns.

  The table editor or other table manipulation tools can be used to populate the rows of the global action table and to add columns as needed.

- Names the global action table SSMV2_GBLEXIT_TBL by default.

  Its name can be changed in the SSMGLOBALEXITTBL parameter. Only one global action table can be active. However, you can switch between multiple global action tables by setting the SSMGLOBALEXITTBL parameter.

There can be multiple actions that match a global event in the global table. The SSM engine assigns a numerical score to each action that matches, so that it can choose the action that most closely matches the event. Actions can be generic, meaning they only specify the event type (ADD, DELETE, or UPDATE), or they can be specific, meaning they specify the generic information and also more specific information, such as the table name being updated. The numerical score increases for each column of the action that matches the corresponding event type. The highest score is the closest match.

The RDF-triggered global events share the global action table that is used by the global BEGIN action. These actions also share the parameters for setting the action table name and enabling the action.

There are a number of variables available to global actions.

More information:

Built-in Variables (see page 220)
Process Events (see page 228)
BEGIN Event (see page 234)

DELETE Event Example

The following example shows how the DELETE event can execute a request rule called SSMDELGX, which is passed in the name and table of the deleted resource. The rule checks whether that deleted resource is a prerequisite for another resource in the table and issues a warning message.

```
SSMV2_GBLEXIT_TBL ----------- TABLE DATA EDITOR ----------- COLUMNS 00001 00124
Command ===> Scroll ===> PAGE
COL--> EXIT_EVENT    EXIT_RES_TABLE    EXIT_RES_NAME    EXIT_ACTION
********************************TOP OF DATA ********************************
000001  DELETE                            RULE("SSMDELGX &SSM!TABLE,&SSM!NAME")
******************************BOTTOM OF DATA********************************
)REQ SSMDELGX
)PROC

/*********************************************************************/
/* Throw away rule name and parse input parameters                   */
/*********************************************************************/
PARSE UPPER ARG . parms
PARSE VAR parms xtable ',' xname

/*********************************************************************/
/* See if deleted resource is a prereq for another resource in this */
/* table                                                             */
/*********************************************************************/
ADDRESS SQL
"SELECT NAME FROM "xtable" WHERE PREREQ IN ('"xname"')"

/*********************************************************************/
/* If a row is returned then issue a warning message                 */
/*********************************************************************/
IF name.0 > 0
THEN DO
   msgtext = 'Resource 'xname' was deleted from table 'xtable' and is',
   'a prereq for 'name.1
   ADDRESS WTO "TEXT('"msgtext"') ROUTE(1)"
END  /* THEN DO */
)END
```

Non-standard and Complex Resource Management

Standard resources managed within the SSM component start through a traditional z/OS START command initialize, and then stay active until it is desired to stop them or the system is being IPLd. Additionally, these standard resources produce specific WTO's that indicate their true current state, such as when it reached its initialization or UP state or when it is being STOPPED. Within SSM, it may be desired to manage the state of resources that do not adhere to the characteristics of these standard types of resources. This section describes the actions needed to manage various non-standard resources within SSM and identify supporting sample rules and programs in which more information can be found.

Manage USS Deamon Server Processes

USS daemon server processes such as INETD and FTP can be managed within SSM by implementing logic as described within the SSMUSS1 sample OPS/REXX program. This sample OPS/REXX program located in the opsmvshlq.CCLXSAMP dsn, is intended to demonstrate an automated technique that can be used within the System State Manager component to monitor and control specific USS processes. The default logic of this sample program is intended to handle USS daemon server processes specifically INETD and/or FTP, but can be used to managing any USS process that behaves in the same manner. Refer to this sample program for complete logic details as well as implementation steps needed to manage these types of resources.

How to Use the Full Capabilities of SSM

The next sections guide you through the steps required to create and implement tables, rules, and procedures that will control your started tasks:

1. Take a snapshot of your system (see page 238)
2. Review and customize the STCTBL table (see page 241)
3. Review and modify the STCTBL_ACT table (see page 242)
4. Auto-enable rules that monitor the started tasks (see page 244)
5. Add STCTBL and STCTBL_ACT tables to the SSM directory table (see page 247)
6. Test the SSM operation (see page 248)
7. Perform an IPL with SSM (see page 252)

Step 1: Take a Snapshot of Your System

The Snapshot function captures data about started tasks on your system and uses it to create a started task resource table that contains a row for each started task that runs on your system. Snapshot also creates an action table, which contains the appropriate actions needed to control the started tasks. Using Snapshot greatly simplifies configuring and maintaining the started task data.

To take a snapshot of your system

1. Choose option 4.11 from the primary OPSVIEW panel.

   The panel displays listing SSM functions as shown here:

```
System State Manager---------- CA99 -- O P S V I E W ---------- Subsystem OPSX
 Date/Time: 2009/04/05 10:31

    1  Control       - Set/Display SSM parameters and resource tables
    2  Status        - Set/Display states of SSM controlled local resources
    3  Snapshot      - Create/Modify a local SSM started task resource table
    4  Scheduler     - Set/Display schedules for SSM controlled resources
    5  Group Manager - Create/Manage groups of SSM resources
    A  Action Editor - Create and maintain SSM action tables
    G  Global Status - Set/Display states of SSM controlled global resources
    R  Resource Edit - Create and maintain SSM resource tables

 Press END to return
```

2. Choose option 3 Snapshot from the main System State Manager panel.

   The following SSM Snapshot panel displays:

```
SSM Snapshot-------------- XE44 -- O P S V I E W -------------- Subsystem OPSX
COMMAND ===>
The Snapshot facility scans the current system for active started tasks and
creates a System State Manager table for them.  This process may take several
minutes to complete.  After the Snapshot table is built, the SQL table editor
will be invoked to allow you to view and customize the resulting table.

SNAPSHOT RULES DATASET:
 Rules DSN ===> 'OPSDEV.O.CCLXRULM'                                (Required)

SNAPSHOT INPUT FROM PARMLIB MEMBER LOAD
 SYSPARM     ===> 00

STC IDENTIFICATION DATASET(S):
 User   DSN  ===>                                                    Opt)
 CA     DSN  ===> 'OPSDEV.O.CCLXCNTL(SNAPDATA)'                     Req)

SNAPSHOT OUTPUT:
 Mode        ===> A ( Add/Refresh/Update )  SSM0 Cols    ===> Y (Y/N)
 Table Name  ===> STCTBL                    SSM version  ===> (Currently 2 only)
Enter SNAPSTART to BEGIN Snapshot, Press PF3 to EXIT without taking Snapshot
```

3. First-time users of SSM should enter the following data:

   - The data set name containing the SNAPRUL1 and SNAPRUL2 rules.

- The name of OPS.CCLXCNTL(SNAPDATA) data set for the CA DSN field.
- A for the Mode field.
- STCTBL for the Resource Table Name field. If you prefer to use a different name for this table, you will have to modify the SSM rules packets. For more information, see Step D: Auto-enable Rules That Monitor Started Tasks in this chapter.
- 2 for the SSM version.
- Y for the SSMO Cols option only if you are running CA Network and Systems Management System Status Manager CA OPS/MVS Option.

4. After you have typed entries into all required fields, enter the SNAPSTART command on the command line.

   The Snapshot process begins. It may take several minutes to survey your system and create the started-task resource and action tables. You will see several messages as the process progresses. Step A is complete when the Snapshot facility invokes the CA OPS/MVS table editor so that you can view and edit the resource table.

You should now have two new tables, the STCTBL resource table and the STCTBL_ACT action table.

Fields on the SSM Snapshot Panel

The following is a list of fields on the SSM Snapshot Panel:

Rules DSN

The full name of the data set containing the SNAPRUL1 and SNAPRUL2 rules. The data set does not have to be a valid rules data set for the active CA OPS/MVS on the system. The rules are read from the specified PDS and installed using the *DYNAMIC rule set.

The SNAPRUL1 and SNAPRUL2 rules are shipped to you in the OPS.CCLXRULM data set.

SYSPARM

The suffix list of IEASYSxx members of the system parmlib that was used to IPL the system. The IEASYSxx members are used to locate other members that are relevant to system task initialization.

The suffix is obtained using the OPSIPL REXX function, and is not modifiable.

User DSN

The full name of the data set that contains user-supplied started task data. The Snapshot facility uses the data to determine the type of a started task, to specify START and STOP commands, and to override some Snapshot actions.

This field is optional. If you do specify a data set name, entries (identified by the TYPE parameter) in this data set override matching entries in the data set supplied by CA. See CA DSN below.

CA DSN

The full name of the data set containing started task data supplied by CA. The Snapshot facility uses the data to determine the type of a started task, to specify START and STOP commands, and to override some Snapshot actions.

As shipped on the CA OPS/MVS distribution media, this data set is the OPS.CCLXCNTL(SNAPDATA) member. Because we maintain the data in the OPS.CCLXCNTL(SNAPDATA) member, do not modify it. Instead, you can copy any entries you want to modify into the User DSN (described above) and make the modifications there.

This field is required.

Mode

The Snapshot execution mode. Valid values are:

ADD

Adds started task data to an existing resource information table without modifying any entries in the table.

If the table that you specify in the Table Name field does not exist, the Snapshot facility creates a new table and places all of the started task data it finds into the new table.

REFRESH

Clears and rebuilds an existing resource information table.

Note: This is the same process that occurs when you specify a value of ADD, and the table in the Table Name field does not exist.

UPDATE

Modifies existing data in the table but does not add any new started tasks to the table.

Resource Table Name

The name of the resource table that is to contain the started task control data. The action table name is assumed to be the resource table name suffixed with _ACT.

SSM version

Specifies SSM version for which required columns should be defined. Currently 2 is the only supported SSM version

SSMO Cols

Specifies whether the required columns for the CA Network and Systems Management System Status Manager CA OPS/MVS Option product should be included when a new SSM resource table is created.

Step 2: Review and Customize the STCTBL Table

After the Snapshot table is built, the SQL table editor is invoked to let you view and customize the resulting table.

Your STCTBL table will resemble the resource table in How SSM Works (see page 273) in this chapter. It will contain data regarding the started tasks that the Snapshot facility found running on your system or contained in the specified or active SYSPARM.

To review and customize the STCTBL table

For each row in STCTBL, perform these steps:

1. Decide if SSM should control this started task. If not, delete the row from the table.
2. Verify that the started task has the correct job name.
3. Verify that the PREREQ column contains the correct prerequisite task names.
4. Modify the TYPE column to fit your environment. Generally, the TYPE value that the Snapshot facility provides will be suitable, but you may want to be more specific. For example, if you have several CICS regions defined in STCTBL, the Snapshot gives each region a type of CICS. You may prefer to group your production regions with a TYPE value of PRODCICS and give your test regions a TYPE value of TESTCICS.
5. When you test SSM operation as discussed in Step F: Testing SSM Operation and Step G: Perform an IPL With SSM in this chapter, your testing environment may prohibit you from testing specified tasks such as VTAM or JES. If this is the case, then change the MODE column value from ACTIVE to PASSIVE for those tasks that you want to leave out of your initial tests.

6. Supply the correct IPL_STATE value that will tell SSM how this task should be handled after an IPL. The IPL_STATE column initiates the DESIRED_STATE column after an IPL. Valid values for the IPL_STATE column are:

   UP

   SSM should start the task at IPL.

   DOWN

   SSM should **not** start the task at IPL.

   IPL

   Something outside of SSM will start the resource, but SSM should manage it after it becomes active.

This step is complete when your STCTBL table is tailored for your environment.

Step 3: Review and Modify Your STCTBL_ACT Table

The STCTBL_ACT table should contain the correct START and STOP commands for the resources in your STCTBL (started task) table.

To review and modify your STCTBL_ACT table

1. From the table editor primary panel (=2.6), press Enter.

   The list of tables for editing displays.

2. Review the default actions that the Snapshot facility creates for the ACTION_TEXT column. These default actions will invoke one of the request rules or OPS/REXX programs described in the following Base Product Components table:

Component	Description
SSMSTATE	A request rule that determines the current state of the started task. This EXEC runs when the current state is UNKNOWN and the desired state is set to some other value. SSMSTATE determines the actual state of the task and sets the current state value accordingly.
STOPCICS	An OPS/REXX program that shuts down CICS regions.
STOPDB2	An OPS/REXX program that shuts down a DB2 region.
STOPJES2	An OPS/REXX program that shuts down JES2.
STOPNETV	An OPS/REXX program that shuts down a NetView region.
STOPSTC	An OPS/REXX program that passes up to three shutdown commands to stop a started task.
STOPVTAM	An OPS/REXX program that shuts down a VTAM region.

Component	Description
SSMUPIPL	A request rule that sets the desired state to UP for those resources that had an IPL_STATE of IPL.

Various SSM related OPS/REXX programs including the above STOPCICS, STOPDB2, STOPJES2, STOPNETV, and STOPSTC are distributed as user modifiable samples located within your installed *hlq*.CCLXSAMP data set.

3. Review the default resource control logic within these SSM samples (as well as other SSM* and SHUT* samples that reside in this same data set) to determine if they are needed within your SSM configuration. If so, copy them into your *hlq*.USER.REXX library.

   SSM issues the basic commands of START *jobname* or STOP *jobname* for all started tasks, and more specific commands will be issued for tasks defined under the ACTION_RES_TYPE column.

   For more information about STOP*xxxx* REXX programs, review the members in the OPS/REXX library.

This step is complete when your STCTBL_ACT table contains the correct START and STOP commands for the resources in your STCTBL (started task) table.

Step 4: Auto-enable Rules That Monitor Started Tasks

You need to enable and auto-enable the rules packets that update the current state for each started task in the STCTBL table. Many of these rules packets are prewritten. You can find them in the OPS.CCLXRULM data set. Each rules packet contains subsystem specific rules that correspond to the data in the TYPE column of your started task table. The names of these rules packets have the format SSM*xxxx*, where *xxxx* is the subsystem name.

Note: For instructions on how to enable and auto-enable rules packets, see Lesson 5: How to Enable and Disable Rules and Rule Sets in the chapter "How to Begin Using the Product."

Some rules packets are composed of multiple rule members. Rules packets use the SQL update instructions to update the STCTBL with the appropriate status.

To auto-enable rules that monitor started tasks

1. If you did not use the recommended name of STCTBL, you will have to change this table name.
2. Create rules for started tasks for which we supply no prewritten rules.
3. Use an existing rules packet as a model, and change the MSGID text to the IDs of messages representing the UP and DOWN states for that task.

   Besides updating the CURRENT_STATE column values to UP or DOWN, rules packets track the starting and stopping of the resources and so prevent duplicate tasks from being started. You do not need to track starting and stopping, but doing so is useful because it enables you to see how far the task has progressed in trying to reach its desired state.
4. Besides enabling the rules packets that monitor the CURRENT_STATE for each started task, you must also enable the following rules members:

   - SSMBEGIN
   - SSMEOM
   - SSMHASP3
   - SSMIEF4
   - SSMSTART
   - SSMSTATE
   - SSMSTOP
   - SSMUPIPL

   These members include the command trapping rules that are needed to intercept manually entered S or P commands.

   The following sample OPS/REXX programs are provided in the OPS.CCLXRULM data set:

- SSMBEGUX is a user exit program called by rule SSMBEGIN, and therefore must exist in the same rule set in order for the rule to be enabled. For information on customizing SSMBEGUX, see section Customizing Startup with SSMBEGUX in this chapter.
- SSMSTAUX is a user exit program called by rule SSMSTATE, and therefore must exist in the same rule set in order for the rule to be enabled. For information on customizing SSMSTAUX, see documentation in the sample SSMSTAUX program.

This step is complete when the required rule members and rules packets for each started task are auto-enabled.

Sample Code

The following is an example of the SSMCICS rules packet:

```
)MSG DFH*
)PROC
/*********************************************************************/
/*                                                                   */
/*   NAME       - SSMCICS                                            */
/*   PURPOSE    - Set the current state of CICS regions based on     */
/*                message traffic.                                   */
/*   RELATED    - STATEMAN                                           */
/*   GLOBALS    - None                                               */
/*   PARMS      - None                                               */
/*   KEYWORDS   - None                                               */
/*   LANGUAGE   - OPS/REXX                                           */
/*   HISTORY    - 27 APR 1993 GEM  - Original implementation         */
/*                                                                   */
/*   NOTES      -                                                    */
/*                                                                   */
/*********************************************************************/
If Opsinfo('EXITTYPE') ^== 'MVS' Then Return
job   = msg.jobname
msgid = msg.id
Select
/*-------------------------------------------------------------------*/
/* DFHSI1500 applid element startup is in progress for CICS ...      */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----*/
  When msgid == 'DFHSI1500' then
    If wordpos('startup is in progress',msg.text) > 0 then
      Address SQL "Update STCTBL set current_state='STARTING'",
                  "where jobname=:job and type='CICS'"
/*-------------------------------------------------------------------*/
/* DFHSI1517 applid Control is being given to CICS                   */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----*/
  When msgid == 'DFHSI1517' then
      If wordpos('Control is being given to CICS',msg.text) > 0 then
        Address SQL "Update STCTBL set current_state='UP'",
                    "where jobname=:job and type='CICS'"
/*-------------------------------------------------------------------*/
/* DFHTM1715 applid product is being quiesced by userid ...          */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----*/
  When msgid == 'DFHTM1715' then
    Address SQL "Update STCTBL set current_state='STOPPING'",
                "where jobname=:job and type='CICS'"
/*-------------------------------------------------------------------*/
/* DFHKE1799 applid TERMINATION OF CICS IS COMPLETE                  */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----*/
  When msgid == 'DFHKE1799' then
    Address SQL "Update STCTBL set current_state='DOWN'",
```

```
                "where jobname=:job and type='CICS'",
                "and current_state <> 'ERROR'"
/*-------------------------------------------------------------------*/
/* Not interested in this message                                    */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+----*/
  Otherwise nop
End
Return
```

Step 5: Add STCTBL and STCTBL_ACT Tables to the Directory Table

The Stateman Controls panel lets you add your tables to the control of the SSM directory table.

To add your STCTBL and STCTBL_ACT tables to the SSM directory table

1. Choose option 1 from the Stateman Controls panel (=4.11).

   The following panel displays:

```
SSM Control--------------- XE44 -- O P S V I E W ------------ Row 1 to 2 of 2
 Command ===>                                                 Scroll ===> CSR
 Date/Time: 2009/07/16 16:39                                    Wait ===> 10
                System ===> *
  Parameters:  Stateman ===> ACTIVE          (Active/Passive/Inactive/Noprereq)
               Statetbl ===> SSM_MANAGED_TBLS
                Version ===> 2
                                                        State Names
 Cmd  Managed Table     Mode  Action Table       Up       Down     Unknown  TNG
 --- ------------------ ---- ------------------ -------- -------- -------- ---
 ADD                     A                     UP       DOWN     UNKNOWN   N
     ZSMQAT1             A   ZSMQAT_ACT        UP       DOWN     UNKNOWN   Y
     ZSMQAT2             A   ZSMQAT_ACT        ONLINE   OFFLINE  UNKNOWN   N
 ****************************** Bottom of data ******************************
```

   This panel displays the current operating mode that SSM is using, the name of the directory table (SSM_MANAGED_TBLS is the default), and a control section.

2. In the control section, add your tables to the control of the SSM.
3. Change data on this panel by tabbing to the Stateman field, type PASSIVE and press Enter.

   This places SSM in PASSIVE mode.

4. Tab to the following fields and type the data shown above in Add Table.
5. After you have placed entries in all fields, press Enter.

This step is complete when the STCTBL_ACT and STCTBL tables are defined in the directory table of the SSM, SSM is in PASSIVE mode, and the current state and desired state match for each system resource defined in the STCTBL table.

You can view and change the states using OPSVIEW option 4.11.2.

Step 6: Test the SSM Operation

In this step, you use the started task resource table, STCTBL and the action table STCTBL_ACT, to test that SSM correctly starts and stops resources. To simplify testing, use the ISPF split screen feature to create two viewing windows:

- In the first window, use the System State Manager Resource Control panel, which is option 4.11.2 in OPSVIEW.
- In the second window, run a product that enables you to view the SYSLOG data set and issue commands, such as the CA SYSVIEW product or the IBM SDSF product.

Test the STATEMAN Rules Packet

To ensure that the rules in the STATEMAN rules packet correctly set the CURRENT_STATE columns in your started task table (STCTBL), do the following:

1. Issue z/OS commands to start and stop the started tasks defined in the table.
2. Verify that the current states are DOWN when the tasks are actually stopped and UP when the tasks are initialized completely.
3. Because SSM is running in PASSIVE mode, check that the desired states change along with the current states.

Test SSM

To determine whether SSM correctly starts and stops your started tasks, do the following:

1. Issue z/OS commands to stop the tasks.
2. Verify that both the current state and desired state are set to down.

   On the System State Manager Control panel, enter ACTIVE in the Stateman field to change the SSM resource monitoring mode to ACTIVE. Doing this causes the SSMBEGIN request rule to execute, prompting you with the messages shown in Setting the Desired State in this chapter. Because you have already set the desired state, select option 5 in response to these messages.

   Note: The WTOR message from the SSMBEGIN request rule will not be sent to your TSO session.
3. Start one of your started tasks by issuing the following z/OS command:

   S *taskname*

   This should cause the SSMSTART rule to capture the S *taskname* command and update the desired state to UP for that resource. SSM then initiates the action defined in the STCTBL_ACT table to get the task started.
4. Check that the task starts correctly. Once it has initialized fully, verify that both the current state and desired state for the task are set to UP.

   While SSM is restarting the task, you can check its progress by locating these messages in the SYSLOG or in the OPSLOG. In this sample, SSMSTART changes the desired state of a job to up.

   ```
   S taskname
   SSMSTART: DESIRED_STATE OF taskname CHANGED TO UP
   OPS7902H STATEMAN ACTION FOR STCTBL.taskname:
   DOWN_UP=OPSCMD COMMAND('START taskname') NOOUTPUT
   OPSCMD COMMAND('START taskname') NOOUTPUT
   START taskname
   ```
5. Issue this command:

   P *taskname*
6. Verify that SSM correctly stops the started task and that both the current state and desired state are set to DOWN after the task has ended. The process SSM uses to stop the task is much the same as the process that executes when you issued the S *taskname* command, except that the messages in the SYSLOG or OPSLOG will reflect the stopping actions.

Test Maintenance of Desired States

To test how well SSM keeps a resource in its desired state, perform these steps:

1. Start a started task and verify that the current state and desired state are set to UP.
2. Simulate the task abending by entering the z/OS command to stop the resource. Use the command STOP *taskname* if the task usually requires a P *taskname* command.

   The STOP command will bypass the command trapping rules, causing the desired state **not** to be set to DOWN.
3. When the task ends, verify that SSM restarts it.

   A restart should take place because initially you set current state and desired state to UP, and then the STOP *taskname* command triggered a rule that changed the current state of the task to DOWN. This mismatch of current state (DOWN) and desired state (UP) should cause SSM to dispatch the action needed to restore the task to its desired state.

Test Prerequisite Checking

To test whether SSM checks prerequisites (if any are defined) for your started tasks, follow a procedure like the one described below.

Suppose that your STCTBL table contains rows defining these tasks:

NAME	DESIRED_STATE	CURRENT_STATE	PREREQ	MISSING_PREREQ
TESTTASK1	DOWN	DOWN	NULL	NULL
TESTTASK2	DOWN	DOWN	TESTTASK1	NULL

TESTTASK1 has no prerequisites but is itself a prerequisite for TESTTASK2, so TESTTASK1 must be active before TESTTASK2 can start.

To verify this

1. Issue the command S TESTTASK2.

   The desired state for TESTTASK2 will change to UP, and the MISSING_PREREQ column value will change to TESTTASK1, indicating that TESTTASK2 cannot start until this prerequisite is satisfied.

2. Next, issue the command S TESTTASK1 and verify that first TESTTASK1 and then TESTTASK2 start.

3. After both tasks become active, issue the command P TESTTASK1 to stop the first task.

   The desired state for TESTTASK1 changes to DOWN and the MISSING_PREREQ value for the task changes to TESTTASK2, indicating that SSM cannot stop TESTTASK1 until the task that requires it. TESTTASK2 is also stopped.

4. Issue the command P TESTTASK2 now.

   SSM will stop TESTTASK2 first, then TESTTASK1.

Step 6 is complete when you have successfully tested the SSMSTART and SSMSTOP rules and the following SSM operations:

- Starting and stopping of started tasks
- Ability to keep a resource in its desired state
- Prerequisite checking

Step 7: Perform an IPL with SSM

You need to test how SSM functions during system shutdown and at IPL time. Before you do so, make sure that:

- The CA OPS/MVS STATEMAN parameter is set to ACTIVE.
- You have successfully completed Steps 1 through 6. If you set the monitoring mode to PASSIVE for started tasks that you did not want to include in your other SSM tests, change the MODE value for those tasks to ACTIVE and repeat Step 6 for each task.
- The MODE value for the STCTBL table and each started task is ACTIVE.
- The current states and desired states in your STCTBL table are UP for tasks that are active or DOWN for inactive tasks.
- You have created a new COMMND*nn* member of the Logical Parmlib Concatenation that contains the CA OPS/MVS START command, the tasks (if any) that SSM will not control, and the tasks that SSM will not start at IPL but will manage after those tasks become active (these tasks will be those for which you set the IPL_STATE value to IPL in Step 2).

Perform a System Shutdown

With SSM, system shutdown is nothing more than setting the DESIRED_STATE to DOWN for all active started tasks that SSM controls. The OPS.CCLXSAMP data set contains a member called SSMSHUT, which provides various options you can tailor to meet your shutdown requirements.

If you intend to use the SSMSHUT program when you implement SSM on your production system, see The SSMSHUT Program in this chapter to understand how it works. For now, invoke SSMSHUT by issuing the following console command:

```
!OI SSMSHUT
```

Note: In the console command above, the ! represents the OSF command character.

In response, you will see the following message:

```
002 SSMSHUT MVS System shutdown requested at 12:20. Reply
  within 00:01 'A' to abort or 'N' to shutdown now.
```

Reply N to initiate the shutdown process. The following occurs:

- The SSMSHUT program sets the DESIRED_STATE column for all started tasks in the STCTBL table to DOWN.
- SSM responds to the mismatch of current state (UP) and desired state (DOWN) by invoking the correct shutdown command or procedure specified in the STCTBL_ACT table.
- The started tasks end and the appropriate rules packets set the current state to DOWN.
- SSMSHUT will enable the SSMSHUTM rule, which will display all resources that have not terminated due to prerequisite constraints every 60 seconds. The SSMSHUTM rule must be present in the active rule sets.

Once you start a system shutdown or IPL, you can invoke the SSMDISP OPS/REXX program. By using this command periodically, you can keep track of what the current state is and what the desired state is as time lapses.

The following variables can help you diagnose any problems with SSM that may occur during your initial system shutdown or IPL.

OI SSMDISP

Displays resources where the current state does not equal the desired state.

!OI SSMDISP ALL

Displays all resources.

For more information, see The SSMDISP Program in this chapter.

The SSMDISP REXX program should display all started tasks that do not have both their current state and desired state set to DOWN. If you have any tasks that do not have a current state of DOWN, then one of the following is true:

- The task ended but the rule did not execute correctly and set the CURRENT_STATE to DOWN. To correct the problem, invoke the STATESET OPS/REXX program using this command:

  `!OI STATESET STCTBL.`*`taskname`* `CURRENT(DOWN)`

- The task has a dependent resource and the MISSING_PREREQ column for the task lists a task that has not ended. SSM will not stop the task until the current state is DOWN for the prerequisite task. Check the actual status of the prerequisite task, or issue the following command to bypass prerequisite checking for the task:

  `!OI STATESET STCTBL.`*`taskname`* `MODE(NOPREREQ)`

- SSM initiated the action to stop the task, but the task is hung.

Proceed with your shutdown of the system and note any tasks that needed special attention.

Perform a System IPL

Perform the IPL using your newly created COMMND*nn* member. Once CA OPS/MVS initializes and SSM becomes active, the SSMBEGIN request rule executes and prompts you to choose a startup option.

Note: Remember that you *must* use a startup option to initiate SSM at IPL time.

At this point, both the CURRENT_STATE and DESIRED_STATE columns in the STCTBL table are set to the table-relative UNKNOWN state. Your STCTBL entries will resemble the following example (not all columns are shown):

NAME	CURRENT_STATE	DESIRED_STATE	IPL_STATE
TASK1	UNKNOWN	UNKNOWN	UP
TASK2	UNKNOWN	UNKNOWN	IPL
TASK3	UNKNOWN	UNKNOWN	DOWN

Reply 3 to the WTOR to initiate the startup process. The following occurs:

1. The SSMBEGIN rules copy the contents of the IPL_STATE column of the STCTBL table into the DESIRED_STATE column. TASK1 now has DESIRED_STATE and IPL_STATE values of UP, TASK2 has DESIRED_STATE and IPL_STATE values of IPL, and TASK3 has DESIRED_STATE and IPL_STATE values of DOWN.
2. A mismatch of the CURRENT_STATE (UNKNOWN) and the DESIRED_STATE (UP, DOWN, or IPL) causes SSM to search the STCTBL_ACT table and dispatch the appropriate start up action. In this case, you invoke the request rule as shown here:

   ```
   RULE("SSMSTATE TABLE(&SSMTABLE) NAME(&NAME) JOBNAME(&JOBNAME)...")
   ```

3. The SSMSTATE request rule detects what actual state the task is in on your system and sets its CURRENT_STATE accordingly. Because this is an IPL, most tasks will be DOWN except for those that SSM does not start. These tasks may be UP.

   The entries in the STCTBL table now look like this:

NAME	CURRENT_STATE	DESIRED_STATE	IPL_STATE
TASK1	DOWN	UP	UP
TASK2	UP	IPL	IPL
TASK3	DOWN	DOWN	DOWN

4. SSM reacts to the mismatched DESIRED_STATE and CURRENT_STATE again, searches the STCTBL_ACT table, and dispatches the correct start command or procedure for each task that has a CURRENT_STATE of DOWN and a DESIRED_STATE of UP. If any tasks have a CURRENT_STATE of UP and a DESIRED_STATE of IPL, the SSMUPIPL request rule changes the DESIRED_STATE of those tasks to UP. SSM takes no action for any task that is not supposed to be started at IPL time. Such tasks have CURRENT_STATE and DESIRED_STATE values of DOWN.
5. Once the started tasks initialize, the SSM rules packets set the CURRENT_STATE columns to UP for each task.

Note: As long as TSO is available, use OPSVIEW option 4.3 to view server activity. This data can help you diagnose any problems you may encounter during system shutdown or IPL by showing server commands in progress and any excessive queue delay times. You can also use OPSVIEW option 4.11.2 (System State Manager Resource Control) to watch the states of resources change.

Create Other Resource and Action Tables

To create additional SSM tables, see the chapter "Editing Relational Tables (see page 445)." You can use the STCTBL and STCTBL_ACT tables you created using the Snapshot facility as a template for other tables, or the Insert option of the table editor will create table models with the columns SSM requires.

To create tables outside of the table editor, you can issue SQL CREATE TABLE statements in a REXX program, from native TSO, or from a CLIST. The sample CREATE TABLE statements shown under Deciding How Many Tables to Define in this chapter list the column requirements for resource and action tables.

Decide How Many Tables to Define

In general, define a separate table for each type of resource instead of mixing different resource types in one table. This approach ensures that most of the columns in each table are specific to a particular resource.

For example, suppose that your company has a remote warehouse that uses a CICS-based inventory application. This type of application requires a cluster controller at the remote site, an SNA communications line, a 3745 communications controller, the CICS region, VTAM, and JES. With all of these resources defined in one table, certain data items required for the communications controller (NCP name, for example) or CICS region (JOBNAME) would be null for other resources and would waste space in the table. Wasted space is the main disadvantage of defining too few tables.

Sample SQL statement creating a resource table

```
SQL CREATE TABLE SSM_TBL1
(NAME CHAR(18) UPPER CASE NULL PRIMARY KEY,
CURRENT_STATE CHAR(8) UPPER CASE DEFAULT 'UNKNOWN',
DESIRED_STATE CHAR(8) UPPER CASE DEFAULT 'UNKNOWN',
MODE CHAR(8) UPPER CASE DEFAULT 'ACTIVE',
PREMODE CHAR(8) UPPER CASE DEFAULT 'ACTIVE',
REFMODE CHAR(8) UPPER CASE DEFAULT 'ACTIVE',
ACTMODE CHAR(8) UPPER CASE DEFAULT 'ACTIVE',
SCHEDMODE CHAR(8) UPPER CASE DEFAULT 'ACTIVE',
JOBNAME CHAR(8) UPPER CASE,
TYPE CHAR(18) UPPER CASE,
CHKPOINT_STATE CHAR(8) UPPER CASE DEFAULT 'UNKNOWN',
IPL_STATE CHAR(8) UPPER CASE DEFAULT 'UNKNOWN',
PREREQ VARCHAR(250) UPPER CASE,
MISSING_PREREQ VARCHAR(250) UPPER CASE,
PREV_STATE CHAR(16) UPPER CASE,
INTERNAL_DATA1 HEX(32),
RESOURCE_TEXT VARCHAR(128), optional
PRIMARY_SYSTEM CHAR(8)) optional
PRIMARY_SYSTEM CHAR(8)) optional
```

Sample SQL statement defining an action table

```
SQL CREATE TABLE SSM_ACTIONS (ACTION_PROCESS CHAR(8) UPPER CASE PRIMARY
KEY DEFAULT 'ACTION',
ACTION_CURRENT CHAR(8) UPPER CASE NULL PRIMARY KEY,
ACTION_DESIRED CHAR(8) UPPER CASE PRIMARY KEY DEFAULT '',
ACTION_MODE CHAR(8) UPPER CASE PRIMARY KEY DEFAULT '',
ACTION_RES_TABLE CHAR(18) UPPER CASE PRIMARY KEY DEFAULT '',
ACTION_RES_TYPE CHAR(18) UPPER CASE PRIMARY KEY DEFAULT '',
ACTION_TEXT VARCHAR(450))
```

Add User Columns to an Existing SSM Table

You may need to insert additional user control columns within an SSM resource table after the initial SSM implementation.

To add additional user columns to an existing SSM table

1. Enter OPSVIEW option 2.6 to list all RDF tables
2. Type an I (Insert) beside the SSM table (using STCTBL as example), tab to the 'New Table' column, and enter a new table name (such as STCTBL_NEW).
3. Press enter from the 'Specify Attributes Panel.'

   The table structure of the current STCTBL that you are modifying displays.

4. Use ISPF edit to insert new column names and their corresponding structures and then press PF3.

   Important! Do not remove or modify any required SSM monitor columns as documented. Do not name any new column with values that may be used as the desired or current state value of a resource.

   The table is saved with the new column names and their corresponding structures.

5. Type a T (Transfer) beside the STCTBL, tab to the 'New Table Column,' and then enter the name of the newly created table, which is STCTBL_NEW in this example.

   All existing data is transferred from the old table to the new table and the new table is ready to be placed in SSM control.

6. Place the new table in SSM control by first removing the existing STCTBL from SSM control using the OPSVIEW option 4.11.1 SSM control panel.

7. Once removed from SSM control, rename the existing STCTBL to a back-up name such as STCTBL_OLD using option R in the RDF table editor (OPSVIEW option 2.6).

8. Rename STCTBL_NEW to STCTBL using the same R option in the RDF table editor.

9. Re-add the new STCTBL back to SSM control. Keep in mind that depending on the user SSM automation that is in place, the SSMBEGIN routine may fire upon the re-adding of the table. You should first verify what the impact of any user code within this routine may perform. The SSMBEGIN rule can be safely disabled prior to re-adding the table using 4.11.1 if the automated logic path is unknown. Upon the re-adding of the table to SSM control, both the CURRENT_STATE and DESIRED_STATE columns are set to UNKNOWN.

   To re-prime these states to their current values:

   a. Set the SSM global MODE column to PASSIVE using the OPSVIEW 4.11.1 panel.

   b. Issue the TSO command 'OPSQL UPDATE STCTBL SET DESIRED_STATE = CHKPOINT_STATE'.

      This sets the DESIRED_STATE column to the value that it was in prior to the table renaming.

   The SSMSTATE (UNKNOWN action) will fire and set the CURRENT_STATE to the true state of the resource (UP or DOWN). After the CURRENT_STATE and DESIRED_STATES are synchronized correctly, the mode of SSM should be set back to ACTIVE.

Parameters That Control SSM Operation

The following CA OPS/MVS parameters directly affect the operation of SSM. These parameters may be set at product initialization and changed at any time. For more detailed information on these parameters, see the *Parameter Reference*.

SSMACTIVEGLOBAL

Sets a globally visible status for the local system and immediately transmits the status to all CA OPS/MVS systems that are connected through MSF.

SSMAUDIT

If set to YES, causes OPS7914T messages to be sent to OPSLOG for every significant change to the resource tables or directory table.

SSMAUXTBLPREFIX

Specifies an auxiliary table name prefix, making that RDF table eligible for change monitoring by the SSM engine.

SSMDEBUG

Causes messages OPS7913T, OPS7914T, and OPS9999T to be sent to OPSLOG.

SSMGLOBALEXITTBL

Specifies a valid 1- to 18-character RDF table name. The specified table is used to process the SSMGLOBALEXITS parameter.

SSMGLOBALEXITS

Determines whether the global exit facility will be active while SSM is executing.

SSMGLVPREFIX

Specifies a global variable name prefix that will be substituted for the & when the string *&.nnnn* is found in an action string.

SSMMONITOREDCOL*n*

Five parameters that each specify a 1- to 18-character RDF table column name. Columns specified here will be actively monitored for changes by SSM.

SSMPLEXNAME

Sets the local system's SSMplex name and then transmits that name to all systems that have an MSF connection to the local system.

SSMPRIORITY

Sets a global priority for the local system and transmits this value to all CA OPS/MVS systems that are connected through MSF.

SSMSUBPREFIX

Specifies an RDF table name prefix that awakens the SSM subtask manager from a wait state.

SSMSUBRULE

Specifies the name of an enabled AOF request rule to be executed by the SSM subtask manager whenever the subtask manager is awakened.

SSMVERSION

Specifies the version of SSM to activate. Currently 2 is the only supported version.

STATEGROUPMAN

Determines whether SSM automatically maintains Group Manager tables.

STATEIGNOREMPRE

Controls whether nonexistent SSM resource prerequisites are considered missing when prerequisite processing is performed by the SSM engine task.

STATEMAN

Controls the overall mode SSM uses to monitor resources. The default monitoring mode is ACTIVE.

STATEMATCHPREFIX

Controls whether SSM prerequisite and subrequisite processing uses the defined UP and DOWN states for a resource table as states that must match exactly or as prefix values that must match only up to the length of the state names.

STATEMAXACTION

Limits the number of actions per minute that SSM can take for any resource that it manages (five is the default). When this limit is exceeded, SSM issues the message OPS7903W STATEMAN ACTION LIMIT EXCEEDED FOR *table.name*.

This allows SSM to detect and correct any repetitive actions for a problem task. You can write a rule triggered by message OPS7903W that updates the CURRENT_STATE value to ERROR, which could then trigger some action to take care of the problem task.

STATEMAXWAIT

Determines how often SSM scans its tables to detect resources with mismatched CURRENT and DESIRED states.

STATETBL

Names the directory table. The default name is SSM_MANAGED_TBLS.

STATETBLLOG

Determines whether messages are produced in OPSLOG when a change is made to the active SSM directory table using the OPSSMTBL command.

Manage Tables with the OPSSMTBL Command

The SSM uses the directory table to manage the resource information table or tables. The OPSSMTBL command allows you to maintain the directory table; for example, you can:

- Add resource information tables to the directory table.
- Delete resource information tables from the directory table.
- Change the operating mode and attributes of resource information tables listed in the directory table.
- Return the names of resource information tables in the directory into CLIST variables or as messages on your TSO terminal.

OPSSMTBL Command Syntax

The following is the syntax for the OPSSMTBL command:

```
OPSSMTBL (keywords)
```

Note: For detailed information about the OPSSMTBL command and its keywords, see OPSSMTBL Command Processor in the chapter "POI Command Processors" in the *Command and Function Reference*.

Associated Variables

The following table shows the CLIST and REXX variables created by the OPSSMTBL LIST command with CMDRESP(CLIST/REXX). The default value for prefix is SSMTBL and *n* is an ascending index number.

Variable Contents	Variable Name if CMDRESP(CLIST)	Variable Name if CMDRESP(REXX)
The number of resource information tables listed in the directory table	*Prefix*	*prefix*.0
The resource information table name	*Prefixn*	*prefix.n*
The resource table mode	*Prefixn*MODE	*prefix*_MODE.*n*
The up-state name	*Prefixn*UP	*prefix*_UP.*n*
The down-state name	*Prefixn*DOWN	*prefix*_DOWN.*n*
The unknown-state name	*Prefixn*UNKNOWN	*prefix*_UNKNOWN.*n*
The action table name	*Prefixn*ACTION	*prefix*_ACTION.*n*

Variable Contents	Variable Name if CMDRESP(CLIST)	Variable Name if CMDRESP(REXX)
The TNGELIGIBLE value	*Prefixn*TNG	*prefix*_TNG.*n*

Modify Table Data with the STATESET Program

After you create resource tables for the use of SSM, you can change the current state, desired state, and MODE values specified for a resource in three ways:

- By invoking an SQL UPDATE statement from a rule or OPS/REXX program. The rule or OPS/REXX program should contain a clause like the following:

  ```
  ADDRESS SQL
  "UPDATE tablename SET columnname = 'state' WHERE NAME = 'name'"
  ```

- By invoking an SQL UPDATE statement from a CLIST or TSO/E REXX program. This CLIST or REXX program should contain a clause like the following:

  ```
  OPSQL UPDATE tablename SET columnname = 'state' WHERE NAME = 'name'
  ```

- By using the STATESET program

Use the STATESET Program

Invoke the STATESET OPS/REXX program to:

- Change the current state, desired state, or mode of a system resource.
- Change the desired state of other resources associated with this resource.
- Specify whether CA OPS/MVS waits, and how long it waits, for the current state of a resource to equal the desired state.
- List the current state (or desired state) of a resource. If you do not specify the CURRENT, DESIRED, PREREQ, or SUBREQ keywords, STATESET lists the current state of the resource.

Note: The state names UP, DOWN, and UNKNOWN can be used for any resource, and will be changed internally to the values specified in the resource directory table entry for the particular resource table in use.

STATESET Syntax

The following is the syntax for the STATESET command:

```
STATESET (keywords)
```

Note: For detailed information about the STATESET program, including its required and optional keywords, see STATESET Program in the chapter "POI Command Processors" in the *Command and Function Reference*.

STATESET Program Examples

The following examples demonstrate ways to use the STATESET program:

- To set the desired state of all resources that depend on VTAM to DOWN (and see a listing of the status of VTAM and all the resources immediately subordinate to it), use this syntax:

  ```
  STATESET VTAM SUBREQ(DOWN)
  ```

- To list the current and desired states, and the mode of VTAM use this syntax:

  ```
  STATESET VTAM
  ```

- To change the desired state of VTAM to DOWN, use this syntax:

  ```
  STATESET VTAM DESIRED(DOWN)
  ```

Invoke the STATESET Program in Various Environments

You can invoke the STATESET program from:

- Another REXX program (in an AOF rule or automation procedure) in either TSO/E REXX or OPS/REXX. Use this format:

  ```
  CALL 'STATESET' resourcename [options]
  ```

- A TSO environment using the CA OPS/MVS OI command. Use this format:

  ```
  OI STATESET resourcename [options]
  ```

- The TSO command line (such as ISPF option 6 or TSO Ready mode). Use this format:

  ```
  STATESET resourcename [options]
  ```

  Important! The library containing the STATESET program must be concatenated to the SYSPROC ddname.

  Note: You can also use this format as a host command in a TSO/E REXX program or CLIST. In any of the above formats, you can specify *tablename.resourcename* instead of *resourcename*. It is more efficient to include *tablename*.

Manage Tables Through OPSVIEW

To view or edit the contents of SSM tables (and all other CA OPS/MVS relational tables), you can use the CA OPS/MVS Relational Data Framework (RDF) table editor of OPSVIEW.

Edit and Browse Tables Through the Table Editor

To edit or browse tables with the RDF table editor, select option 2.6 from the OPSVIEW Primary Options Menu. CA OPS/MVS responds by displaying the RDF Table Editor Primary Panel, as shown:

```
RDF Table Editor --------------- Primary Panel -------------------------------

OPTION ===>
       B - Browse table                R - Rename table
       C - Copy table                  D - Delete table
       E - Edit table                  F - Free table
       I - Insert new table            T - Transfer table contents

                    blank - Display table list
SPECIFY RELATIONAL TABLE (see note below):
   NAME     ===> _____________________________  (Required for B, C,D,E,F,R,T)
   NEWNAME  ===> _____________________________  (Required for C,I,R,T)
CONFIRM DELETES: YES (Enter YES to require delete confirmation)
NOTE: To use a table on another system specify the table name as system>table
      Specify ? as the system name to get a list of all systems.
```

Edit or Browse Through the System State Manager Interface

To edit or browse tables from the OPSVIEW System State Manager interface, select option 4.11 from the OPSVIEW Primary Options Menu.

CA OPS/MVS responds by displaying the System State Manager menu panel, as shown:

```
System State Manager---------- S034 -- O P S V I E W ---------- Subsystem OPSD
OPTION  ===>
Date/Time: 2009/10/13 08:54
   1  Control       - Set/Display SSM parameters and resource tables
   2  Status        - Set/Display states of SSM controlled resources
   3  Snapshot      - Create/Modify a local SSM started task resource table
   4  Scheduler     - Set/Display schedules for SSM controlled resources
   5  Group Manager - Create/Manage groups of System State Manager resources
   A  Action Editor - Create and maintain SSM action tables
   G  Global Status - Set/Display states of SSM controlled global resources
   R  Resource Edit - Create and maintain SSM resource tables
Press END to return
```

Use the SSM Control Panel

From the SSM Control panel, you can perform these tasks:

- Activate or deactivate the SSM or operate it in PASSIVE mode.
- Specify a new directory table.
- Add a resource information table to the directory table.
- Associate an action table with a resource information table.
- Activate or deactivate monitoring for a resource information table or place the table in PASSIVE mode.

Note: For more information about the OPSVIEW SSM Control option as well as all the options available under the OPSVIEW System State Manager facility, see the *OPSVIEW User Guide*.

SSMDISP Command—Display Resource Status

The SSMDISP sample OPS/REXX program located in your opsmvshlq.CCLXSAMP library, displays the status of SSM resources using a multi-line WTO. To utilize this sample program, copy it into your opsmvshlq.USER.REXX library, or a valid user REXX library that is allocated to your OPSMAIN, and OPSOSF procedures.

You can filter the resources you wish to display by mode or whether the current state is equal to the desired state. Also, you can direct the output of the program to a specific console or *areaid*. If you use the SSMDISP program in conjunction with the SSMDISPC command rule, an OSF server is not required to run SSMDISP. You can call SSMDISP as a subroutine from any OPS/REXX program.

This command has the following syntax:

```
SSMDISP
  [EXC|ALL]
  [MODE(ssmmode)]
  [CNNAME(console name)]
  [AREAID(display area)]
```

The keywords of the SSMDISP command have these values:

EXC|ALL

(Optional) One of the following:

- EXC-Displays only the SSM resources whose current state does not match the desired state; this is the default.
- ALL-Displays all SSM resources, regardless of their state.

MODE(*ssmmode*)

(Optional) Displays only the SSM resources whose mode matches the specified mode. Valid values are ACTIVE, INACTIVE, PASSIVE, and NOPREREQ. The default is to not filter by mode.

CNNAME(*console name*)

(Optional) Specifies the name of the z/OS console to which the multi-line output is directed. The default is MSTRINFO routing with no specific console.

AREAID(*display area*)

(Optional) Specifies the z/OS console display area in which the multi-line output appears if the output is directed to a console through CNNAME or CNID. The default is Z.

Output from SSMDISP

The SSMDISP multi-line WTO output consists of a control line that indicates the overall status of SSM, output column labels, and one or more resource lines. As shown in the following example, the resource lines include the *table.resource name*, mode, current and desired states, and a portion of the missing prerequisite column:

```
OPX1371I Stateman Status: ACTIVE
Resource Name          Mod Current  Desired  Missing Prerequisites
---------------------- --- -------- -------- ------------------------
DGTBL1.DG1             ACT DOWN     UP       DG2
DGTBL1.DG2             ACT DOWN     UP
```

Examples of SSMDISP

The following examples demonstrate ways to use the SSMDISP program:

- To display the status of all active resources whose current state does not match the desired state on all consoles that display MSTRINFO routed messages, use this syntax:

  ```
  !OI SSMDISP MODE(ACTIVE)
  ```

- The following example assumes that the SSMDISPC rule is enabled, intercepts the command, and calls SSMDISP as a subroutine:

  ```
  SSMDISPC ALL
  ```

  The resulting multi-line WTO output is automatically routed back to the originating console. In this case, all resources are displayed, regardless of their state.

SSMSHUT Command—Set Resource State to Down

As part of a general shutdown of resources, usually in preparation for an IPL, the SSMSHUT sample OPS/REXX program located in your opsmvshlq.CCLXSAMP library, sets the desired state of all SSM resources to DOWN. To utilize this sample program, copy it into your opsmvshlq.USER.REXX library or a valid user REXX library that is allocated to your OPSMAIN and OPSOSF procedures.

The progress of the shutdown and any bottlenecked resources are periodically displayed by the SSMSHUTM rule through a multi-line WTO. The SSMSHUT program enables the SSMSHUTM rule when the shutdown begins. The AT and IN keywords of the SSMSHUT command allow you to initiate the shutdown at any time in 23 hours of the current time. The SSMSHUT program also provides the operator with an opportunity to cancel the shutdown.

The EXCLUDE keyword of the SSMSHUT command allows you to perform a partial shutdown of resources by providing a list of the resources (explicit resource names, global variable names, or both) you want to exclude. SSMGA inactive copies of movable resources are automatically excluded when SSMGA is active.

This command has the following syntax:

```
SSMSHUT
  [AT(hh:mm)|IN(hh:mm)]
  [WARNTIME(hh:mm)]
  [CONFIRM(Y|N)]
  [MONITOR(Y|N)]
  [EXCLUDE(resource list|glv names)]
```

Note: The SSMSHUT command uses dynamic time rules to reschedule itself periodically to avoid exceeding the OSFWAIT and OSFRUN time limits in the server.

The keywords of the SSMSHUT command have these values:

AT(hh:mm)|IN(hh:mm)

(Optional) One of the following:

AT-The time at which the shutdown is to occur

IN-The amount of time from the current time at which the shutdown is to occur

The maximum value is 23:00. The default is IN(00:01) if AT is not specified.

WARNTIME(*hh:mm*)

(Optional) The amount of time prior to the actual shutdown time specified by the AT or IN keyword that a z/OS SEND command will be issued to notify users about the impending shutdown. A WTOR will also be issued, allowing the operator to cancel the shutdown if desired. The default is 00:10.

CONFIRM(Y|N)

(Optional) Specifies whether a WTOR is issued to the operator for confirmation that a system shutdown is desired at the specified time. A reply of U allows the shutdown to continue. The default is Y.

MONITOR(Y|N)

(Optional) Specifies whether the SSMSHUTM TOD rule should be enabled to monitor resource shutdown progress once the shutdown has been started. The default is Y.

EXCLUDE(*resource list*|*glv names*)

(Optional) A string of resource names specified as *name* or *table.name*. Global variable names may also appear in the list of names. The values of the variables will be resolved and added to the resource name string.

Example: SSMSHUT

The following example illustrates changing the desired state of all SSM resources except JES2 to the DOWN state in 30 minutes from the current time. It also issues a warning message to all TSO users five minutes before the shutdown and provides the operator with an opportunity to cancel the shutdown.

```
!OI SSMSHUT IN(00:30) EXCLUDE(STCTBL.JES2) WARNTIME(00:05)
```

Chapter 8: Using SSM Global Application

This section contains the following topics:

About SSMGA

The System State Manager Global Application (SSMGA) uses some basic features of SSM, and a new REXX-coded application, to implement monitoring and control of operations that span multiple systems. Using SSMGA, you can:

- Apply cross-system prerequisite and subrequisite to resource status determination.
- Move a resource, or group of resources, from one processor to another with minimum transition time.
- Monitor the status of resources from multiple systems on a single, integrated ISPF display.
- Display and control SSMGA from an operator console or automation procedure by using a command rule.
- Restore the operation of a resource automatically, when its preferred host fails, by restarting the resource on an alternate host.

Sharing Resource Status Information

Two or more systems can share resource status information if they have the same SSMPLEXNAME setting. The SSMplex may include the same systems as a SYSplex, or may include a subset or superset of systems in a SYSplex.

Shared data is maintained in the GST (Global Status Table), which is maintained by the global system. Only one global system is active at any time.

Every SSMplex member has an SSMPRIORITY value between 0 and 999 that is used to select the “global” system.

- Priority 0 systems can never be assigned as the global system.
- Priority 1 is the last priority that will be chosen.
- Priority 999 is the first to be chosen.
- Priority 1000 is used to force an eligible alternate global system to become the global system. The system’s normal priority is added to 1000 to create a priority that is higher than any other normal priority. Once the new system becomes the new global system, the priority is reset to the original priority by subtracting 1000.

Priority affects the choice of a new global system when no global system is active, but otherwise has no effect. See the SSMPRIORITY section in this chapter for more information.

Only systems that have a multi-system facility (MSF) connection to every other system in an SSMplex are eligible to be the global system. Eligibility and priority are both taken into account when selecting a new global system.

In normal operation, the global system uses MSF connections to collect status information and also distribute commands to the other systems.

SSMGA data collection is performed asynchronously by multiple subtasks. Using subtasks for data collection:

- Enables SSMGA global processing to continue, using the most recent data available, even if data collection is delayed on one or more systems in the SSMplex.
- Avoids interference with existing SSM processing.

How SSMGA Works

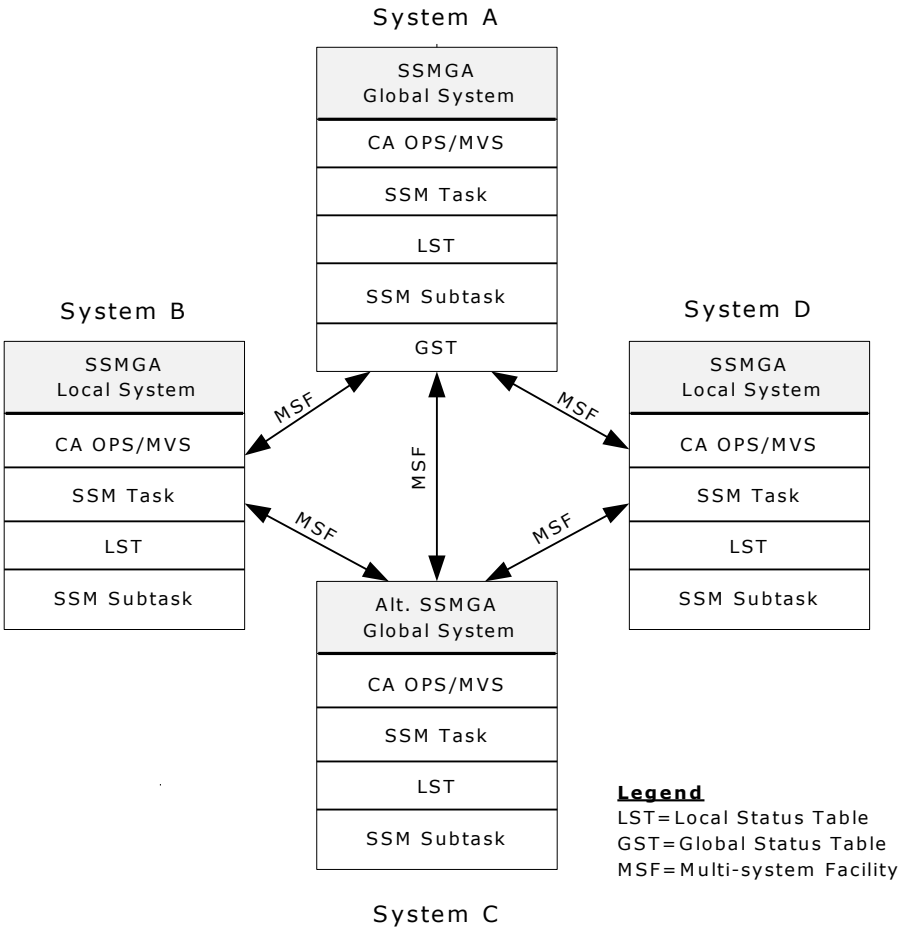
The global system in the SSMplex does the following:

- Maintains its own consolidated local resource status in a single RDF table called the LST (local status table)
- Moves resources
- Maintains the consolidated resource status of all local systems in another RDF table called the GST (global status table)
- Examines changes in the prerequisite and group names resource columns
- Processes the SSMGA commands generated by the SSMGAOPR command rule and queued to the global command RDF table for execution.

Important: Resources defined to Schedule Manager cannot be defined as movable resources to SSMGA.

The advantages of global status management are a high level of multi-tasking and minimal data transfer between systems. Recovery from a global system failure is reduced to selecting a new global system and rebuilding the GST from the remaining systems LST tables.

Equation 1: Shows how SSMGA works.



Local Status Table (LST)

The LST table is maintained by the SSM global event actions that execute when one or more monitored column changes occur in an SSM resource.

- The global event action request rule, SSMGALCL, updates and timestamps resource status information in the LST.
- A system status record is also maintained in the LST with the highest timestamp of any local resource status record in the LST. The highest system timestamp is referred to as the local update time (LUTIME).
- When a resource status change occurs in the LST, the SSM subtask is awakened and transmits the LST system record information to the global system.

Moving Resources

Preventing duplicate starts of the resources when eligible systems fail or are restarted requires special handling of the movable resources by SSMGA; such as:

- Lets the resource ownership take place after every eligible system knows the status of the movable resource in the SSMplex.
- Replicates movable resource definitions, rules, and action table actions on all systems on which the resource can run.
- Determines if a resource is active on the system by using the global event exit for the LST processing.
- Continuous updating of the statuses of the inactive copies by the global system.
- When no system owns the resource, the global system assigns the resource to the first available system in the eligible system list or the system waits for a specific system to join or rejoin the SSMplex.
- Moves the resource back to its home system by using of the optional AUTOHOME functionality once the home system becomes active.

Global Status Table (GST)

In addition to the LST, the global system maintains the consolidated resource status of all the local systems, including itself, in another RDF table called the GST (global status table). The structure of the LST and GST are identical.

- The SSM subtask invokes the SSMGAGBL request rule to perform both global and local system processing, as required.
- On the designated global system:
  - SSMGAGBL processes commands queued in the global command table. Command processing is suspended during the initial build or rebuild of the GST in order to prevent premature resource movement commands.
  - SSMGAGBL examines the global copies of the LST system records.
  - If a system record LUTIME is higher than the maximum global read timestamp previously processed for the system (GRTIME), another subtask is attached to read and process all the LST records whose LUTIME is greater than GRTIME for the system. This insures no global system processing delays by any one local system communication problem.

PREREQ and SSM#GRPLST Resource Column Changes

In addition to updating the GST with LST status information, the system specific subtask examines the PREREQ and SSM#GRPLST resource columns for changes.

- When a cross-system prerequisite specification is detected, a remote status record for both the prereq and subreq relationship between the resources is created or updated.
- These remote records are later read into the LST of the affected local systems for use by the SSM XPREREQ and XSUBREQ process events that are required to evaluate cross-system resource relationships.
- A group membership table (GRT) record is maintained for each defined group name for a resource.
- The system subtask tracks the maximum LST LUTIME processed, which becomes the new GRTIME value in the system record.
- The maximum global status update timestamp (GUTIME) is also recorded in the system record.
- The system subtask updates the LST system record timestamps on the specific system that it processed.

Once the system record is updated in the LST by the global system, the local system processing phase of SSMGAGBL running in the SSM subtask occurs. If the LST system record has a GUTIME that is greater than the maximum GUTIME previously read by local processing (LRTIME) then local processing will read any updated remote status records and any movable resource local records whose GUTIME is greater than LRTIME. Remote status records only exist in the LST. Movable local resource status is propagated to the actual SSM resource records. Non-movable local resource status is never read back from the GST. The highest GUTIME processed by local system processing becomes the new system record LRTIME.

SSMGA Setup Requirements

The SSMGA setup requirements for OPS/MVS parameter settings, auto-enabled rules, REXX programs, resource table columns, and action table entries are discussed in this section. These requirements apply to all systems that will participate in the global SSMplex.

To setup one system, follow Steps 1-8. To replicate these changes to external systems, follow Step 9.

Step 1: Activate the SSM Global Event Facility

In this step you activate the SSM global event facility, which SSMGA uses to process the LST. If you are already using the SSM global event facility, then you can skip this step.

To activate the global event facility, specify the following parameters:

```
SSMGLOBALEXITS=YES
SSMGLOBALEXITTBL=RDF table name
```

Suggested RDF table name value (Default): SSMV2_GBLEXIT_TBL

The specified RDF table name is the name of the global events action table. SSM automatically creates it when parameter SSMGLOBALEXITS is set to YES. For information on manually creating the table before setting parameter SSMGLOBALEXITS, see the chapter "Using System State Manager (see page 177)."

Step 2: Deactivate SSM Processing

Before making any more changes to an existing SSM configuration, it is strongly recommended that you deactivate SSM processing by setting the OPS/MVS parameter STATEMAN as follows:

```
STATEMAN=INACTIVE
```

Step 3: Update the SSM Global Event Facility

SSMGA uses the SSM global event facility activated in Step 1 to maintain a copy of all local resource status information in the LST. In this step, you update the SSM global events action table specified by parameter SSMGLOBALEXITTBL.

The BEGIN, UPDATE, ADD, and DELETE global events in the global events action table must have actions that invoke the SSMGALCL request rule. The action text for each event name should be as follows. If the global event facility is already in use, then add the action text to the end of the existing action text separated by a semicolon.

```
ADD      …   RULE("SSMGALCL &SSM!PROCESS &SSM!RESNAME")
BEGIN    …   RULE("SSMGALCL &SSM!PROCESS IPL(&SSM!IPL)")
DELETE   …   RULE("SSMGALCL &SSM!PROCESS &SSM!RESNAME")
UPDATE   …   RULE("SSMGALCL &SSM!PROCESS &SSM!RESNAME
             COLUMN(&SSM!COLUMN) USERDATA(&SSM!USERDATA)")
```

Note: (…) represents empty key values in the action table.

The following list describes the global events:

- **ADD—**Reads all the resources in the resource table, checks for the existence of optional columns, and calls the ADD status routine to insert or update the resource status record in the LST.
- **BEGIN—**Deletes the LST table and then reads the SSM directory table using OPSSMTBL. Each resource table is processed as an ADD table global event.
- **DELETE—**Updates all LST resource status records for a resource table with a delete flag and update time for global system cleanup processing. The monitored column indicators in the RDF column table for the resource table are removed.
- **UPDATE—**Updates all LST resource status records for a resource table by repeatedly calling the update status routine.

Step 4: Set the Parameters

To adjust the CA OPS/MVS parameter settings.

Follow these steps:

1. Assign an SSMPLEXNAME and SSMPRIORITY to each system that participates in an SSMGA global status management complex.

   `SSMPLEXNAME=SSMplex` *`name`*

   SSMplex name (eight chars max) is the collective name for all systems in an SSMplex. To be an SSMplex member, a system must have an MSF connection to at least one other system, and both must have the same SSMPLEXNAME.

   `SSMPRIORITY=`*`priority value`*

   The Global selection priority (0-999).

   A priority of zero prevents a system from becoming a global system. If no global system is active or the global system goes offline, the eligible system having the highest SSMPRIORITY becomes the global system. For a system to be eligible, it must have an MSF connection to all other systems in the SSMplex, and must be operating normally. If after a global system is selected, a higher priority system becomes eligible to be the global system, the global system does not change automatically. To select any eligible system as the new global system, use the special priority of 1000. SSMGA makes the system that you set to priority 1000 the new global system, and then restore the original priority of the new global system.

   Once you move a global system, the new system gathers data from the other systems. During this time, it is normal for the SETSYS processing display to show two global systems briefly in OPSLOG.

2. Activate the subtask for SSMGA. SSM allows the creation of a separate SSM subtask whose primary function is to perform asynchronous operations on behalf of the primary SSM task.

   To activate the subtask for SSMGA, specify the following parameters:

   `SSMSUBRULE=SSMGAGBL`

   `SSMSUBPREFIX=RDF table name prefix (6 chars max)`

   The Suggested RDF table name prefix: 'SSMGA_'

The subtask is activated by specifying a request rule name other than 'NONE' for the SSMSUBRULE parameter. Each time the SSM subtask is posted, the specified request rule is invoked. The SSMSUBPREFIX specifies an RDF table name prefix that results in a post of the SSM subtask whenever a prefix matching table modification is made by any task other than SSM and the SSM subtask. In addition, a POST(SSMSUB) operand is available on the OPSSMTBL POI command for on-demand posting of the SSM subtask from any environment without a table modification.

Note: Note: Leave the SSMSUBRULE parameter set to NONE (default) until all other changes to an existing SSM configuration are made. Then set the STATEMAN parameter back to ACTIVE. See Step 10.

SSMGA creates the following GLOBAL TEMPORARY tables using the SSMSUBPREFIX value specified:

- LST - local status table on all systems

  `prefix||#L#_STATUS`

- GST - global status table on the global system

  `prefix||#G#_STATUS`

- GRT - global group membership table on the global system.

  `prefix||#G#_GROUPS`

- GCT - global command table on the global system.

  `prefix||#G#_CMDS`

3. Increase the value of the AOFSIZE parameter.

   The AOFSIZE parameter determines the size of the AOF workspace, which is used to store REXX variables. You can increase the value of the AOFSIZE parameter, depending on its current value and the number of SSM resources defined. Error message OPS0998E indicates that the value of parameter AOFSIZE to increase.

4. Increase the value of the GLOBALTEMPMAX parameter.

   The GLOBALTEMPMAX parameter determines the maximum number of *temporary* global variables that OPS/MVS can create. Temporary global variables are used to store the RDF tables. You can increase the value of the GLOBALTEMPMAX parameter, depending on its current value and the number of SSM resources defined. Error message OPS7593O indicates that the value of parameter GLOBALTEMPMAX to increase.

5. Increase the value of the SQLPOOLSIZE parameter.

   The SQLPOOLSIZE parameter determines the size of the storage pool that is used by the RDF to contain SQL data. You can increase the value of the SQLPOOLSIZE parameter, depending on its current value and the number of SSM resources defined. Error message OPS3032T indicates that the value of parameter SQLPOOLSIZE to increase.

6. Increase the value of the MSFPOOLSIZE parameter.

   The MSFPOOLSIZE parameter determines the size of the storage pool that is used by MSF to contain data being passed between CA OPS/MVS systems through MSF. You can increase the value of the SQLPOOLSIZE parameter, depending on its current value and the number of SSM resources defined. Error message OPS3032T indicates that the value of parameter MSFPOOLSIZE to increase.

7. Activate the AUTOHOME functionality.

   Use the parameter SSMAUTOHOME=YES to enable the functionality to move resources back to their home system once this system becomes active.

8. Activate the prerequisite checking.

   Use the parameter SSMGAPREREQCHK=YES to enable the premove prerequisite validation for an initial assignment of a resource at initialization, system recovery, and resource movement commands.

   NOTE: Use the keyword PRECHK for movement commands to explicitly prevent the move when missing prerequisites found.

Step 5: Auto-Enable Rules

In addition to the normal SSM resource status rules, SSMGA execution requires that the following AOF rules be enabled prior to SSM initialization:

SSMGAATH

Performs the AUTOHOME functionality.

This message rule detects the startup of SSMGA and enables the dynamic TOD Rule which executes the AUTOHOME procedure by starting the SSMGAATH OPS/REXX program.

SSMGACCI

Notifies SSMGA of a change in an MSF connection status.

This message rule posts the SSM subtask when the status of an MSF CCI connection changes to ACTIVE or INACTIVE.

SSMGACOM

Notifies SSMGA of a potential MSF connection failure.

This message rule posts the SSM subtask when the monitor task issues the OPS3440O message.

SSMGAEND

Notifies the SSMGA global system that CA OPS/MVS is terminating.

If MSF is already terminated, the early notification may not complete and the global system will know of the termination when its own MSF link with the terminating system goes inactive. This message rule also deletes any outstanding WTOR's issued by the local SSMGA application that may be in a wait state.

SSMGAGBL

Performs both global and local resource processing.

The SSM subtask invokes this request rule. True global processing is only performed when the system is the designated global system. Local processing consists of reading new or updated resource status records for the local system and updating the local status table and replicated movable SSM resources.

SSMGALCL

Keeps the local status table synchronized with the SSM resource table changes.

The SSM global events invokes this request rule. Changed LST records will ultimately be reflected in the global status table and selectively distributed to the LSTs of other systems dependent on the status of cross-system prereq/subreq resources.

SSMGAMSF

Notifies SSMGA of a change in an MSF connection status.

This message rule posts the SSM subtask when the status of an MSF APPC connection changes to ACTIVE or INACTIVE.

Note: When enabled, this rule changes the severity of message OPS3504I to J.

SSMGAMSR

Executes the SUBREQ process events for subreq evaluation.

This request rule checks for moveable resources running on a remote system and removes them from the MISSING_PREREQ column of local resources.

SSMGAOPR

Directs command replies back to the console.

This command rule is the operator interface for queries and commands directed to the global system. When a command is issued from a real or extended console the command replies are directed back to this console. Otherwise, replies are issued as generic WTOs.

The following commands are supported by this command rule.

DPLEX

Displays all SSMGA SSMplex names whose global systems are MSF connected to the system issuing this command. This command determines which SSMplex names can be used in other SSMGA commands that are sent to a SSMGA global system for execution.

DSYSTEMS *ssmplex*

Displays the status of all systems that are in the specified SSMplex.

DGROUPS *ssmplex grpname* [MEMBERS]

Displays the number of resources that are members of the specified group name. In addition, lists the members of that group when the optional keyword MEMBERS is present. Member names appear in the format system.subsys.table.name

DRES *ssmplex resname* [EXC MOVABLE GROUPS SYSLST]

Displays status information for the resource name specified.

MRES *ssmplex resname* TOSYS(*sysname*) DESIRED(*desired state*)

Moves the movable resource to a new system and sets the desired state to designated value.

SSYS *ssmplex sysname* [GLOBAL PLEXNAME(*ssmplex*) PRIORITY(0-999)]

Changes the OPS/MVS SSMGA parameter values on the indicated system.

- PRIORITY changes the global priority value of the system.
- PLEXNAME changes the SSMplex name to a new name.
- GLOBAL forces the indicated system to become the SSMGA global system provided the priority is greater than zero.

MGROUP *ssmplex groupname* TOSYS(*sysname*) DESIRED(desired state)

Moves the movable resources in the specified group to a new system and sets the desired state to designated value.

UGROUP *ssmplex groupname* START/STOP

Starts or Stops all of the resources in the specified group.

URES *ssmplex resname*

```
[MODE(I/P/A/N) PREMODE(A/P/S/I) REFMODE(A/P/I/S)
               ACTMODE(action mode) SCHMODE(I/A)
               CURRENT(current state) DESIRED(desired state)]
```

Updates the resource with the corresponding new column values.

Note: The SCHMODE(I/A) parameter of URES lets you set the value of the SCHEDMODE column for a specific resource. If the value is set to inactive, then the Schedule Manager reset processing bypasses any updates to the DESIRED_STATE column of this resource. For more information, see the chapter "Using Schedule Manager."

HELP command

Displays the syntax for the indicated command

Syntax Notes:

- ssmplex is a valid SSMplex name or * that indicates the SSMplex to which the current system belongs.
- sysname may be in the form system.subsys, system, or *. Omitted portions of the sysname will be completed with values for the current system.
- resname may be in the form system.subsys.table.name, table.name, or just name. A full resource name will only update or display one resource. A partial resource name may update or display multiple resources.

SSMGAPRX

Executes the XPREREQ and XSUBREQ process events for cross-system prereq/subreq evaluation.

This request rule uses the remote resource status records created and transmitted by the global system to the LST of the local system in order to evaluate each xprereq/xsubreq relationship. The global system determines the required remote status records by analyzing the prereq column specifications of each local resource on every system in the group.

Required REXX Programs

Most of the SSMGA rules are simple shells that adjust the parameter list and call another REXX program that actually contains the logic to be performed. This technique avoids the requirement to account for the request rule name as the first parameter in the REXX calling sequence.

The required REXX programs are:

SSMGAATH

Performs the AUTOHOME functionality. Sends resources back to their home systems when the original home system becomes active and reconnects to the SSMGA SSMPLEX.

SSMGAGCM

Displays global data and updates the local resources. This REXX program is attached by SSMGAGST.

SSMGAGST

Processes the GST and LST table. The SSM subtask program is called by the SSMGAGBL request rule.

SSMGALST

Processes SSM global events. The SSM global event program is called by the SSMGALCL request rule.

SSMGAPCK

Performs the premove prerequisite checking to avoid moving resources to the system where the prerequisites are not satisfied. If this system not found resource will be moved to the first system in an alternate system list. This REXX program is called by SSMGASYS to perform validation logic for system recovery and by SSMGAGCM for resource movement commands. To enable this logic, set SSMGAPREREQCHK=YES. This validation works for non-moveable prerequisites only.

SSMGAPRE

Evaluates XPREREQ and XSUBREQ process events. The SSM xprereq/xsubreq evaluation program is called by the SSMGAPRX request rule from the action table. It also supports the user-defined *SYS.AFF.table.name prereq specification for insuring that a movable resource prereq is actually running on the same system as the resource with the prereq specification.

SSMGASYS

Contains the core logic of the SSMGA application. This REXX program is attached by SSMGAGST to process LST and GST changes for a single system in a separate subtask.

SSMGATRM

Contains the OPS/MVS termination global system notification program called by request rule SSMGAEND.

Step 6: Add SSM Resource Table Columns

SSMGA requires several new columns in SSM resource tables to support the functionality provided by SSMGA. Use REXX program OPSSM2CV to add the required SSMGA columns to SSM resource tables by specifying a parameter of SSMGA(Y).

Strictly local resources that cannot be moved to another system do not require any column changes. Defaults for non-existent columns are supplied for the LST and GST tables.

All system name values in SSMGA are in the form:

sysname.subsys

sysname

Specifies the real z/OS system name.

subsys

Specifies the OPS/MVS subsystem name on the system (usually OPSS).

Movable resources require the following new columns:

```
PRIMARY_SYSTEM  CHAR(14)      UPPER CASE DEFAULT('*.*')
SSM#CURSYS      CHAR(14)      UPPER_CASE DEFAULT('*.*')
SSM#DESSYS      CHAR(14)      UPPER CASE  DEFAULT('*.*')
SSM#MOVMOD      CHAR(8)       UPPER CASE  DEFAULT('INACTIVE')
SSM#SYSLST      VARCHAR(350)  UPPER CASE
SSM#GRPLST      VARCHAR(350)  UPPER CASE
AUTOHOME        CHAR(1)       UPPER CASE  DEFAULT('N')
```

The following list describes the new columns:

- PRIMARY_SYSTEM

  Specifies the preferred system to run the resource and is also the constant cross-system prereq specification name for resources that use the movable resource as a prerequisite but do not have a local SSM replicated copy of the movable resource. Even when the movable resource is running on an alternate system, the primary system name is still the cross-system prerequisite name; that is, system.subsys.table.name.

- SSM#CURSYS

  Contains the current system that owns the resource.

- SSM#_DESSYS

  Specifies the system name to which the resource should be moved. Changing the desired system to an alternate system name on the current system copy of the resource will cause the resource to stop on the current system and move to the alternate system.

- SSM#MOVMOD

Controls how the current system is selected. A value other than INACTIVE indicates that the resource is movable from the primary system to any other system specified in the SSM#SYSLST column. The alternate systems are specified as word or comma delimited list. System names are in sysname.subsys format. The sequence of alternate system names in the list is an implied priority of alternate systems for automatic selection of an alternate system.

SSMGA recognizes three special values for SSM#MOVMOD:

- AUTO - Causes SSMGA to automatically move a resource to the next available system in the system list hierarchy when the recovery option is chosen for a detected movable resource owning system failure.
- WAITSYS - Causes SSMGA to wait to assign ownership of a movable resource until the current desired system rejoins the SSMGA plex.
- WAITPRI - Causes SSMGA to wait to assign ownership of a movable resource until the primary system joins the SSMGA plex.

Any other value of move mode will allow movement of resources by manually changing the desired system to an alternate system.

- SSM#SYSLST

  Lists all of the available alternate systems to which you can move data.

- SSM#GRPLST (Optional)

  Specifies SSM resources, such as a workload, global status, or disaster recovery eligibility, using arbitrary 1-18 character names. The ISPF monitoring tool uses the group names in SSMGA to move groups of resources and in the operator command facility for query and action commands. The list of group names is word or comma delimited. SSMGA monitors the resource group name list for changes

- AUTOHOME (Optional)

  Specifies the Yes/No value that determines if the resource can move back to its home system once this system becomes active.

Step 7: Enter Resource Configuration Values

Once the required columns have been added to the resource tables, enter the column data values.

- Enter values for non-movable resources
  - Leave the primary system, current system and desired system columns at their default values of *.* which means the current system.
  - Ensure that the SSM#MOVMOD column contains the value INACTIVE.
  - Ensure that the system list column is empty.
- Enter values for all movable resources:
  - Enter the primary system name with a valid system.subsys name.

    The combination of primary system and resource table names must be unique within the SSMplex. All duplicate names are treated as replicated resources of one active resource.
  - Set the current and desired systems to the primary system name.
  - Set the ACTMODE mode to ACTIVE on the primary system. Set the ACTMODE in INACTIVE on all other systems.

    Note: The resource MODE should always be ACTIVE. SSMGA uses the ACTMODE column to control moveable resources. The ACTMODE will be set to ACTIVE on the system the resource is running on and set to INACTIVE on all other systems.
  - Set the SSM#MOVMOD column value to the appropriate resource recovery value for normal operations.
  - Populate the alternate system list column with the desired system names specified in the order in which they are to be selected.
- Assign optional group names to both movable and non-movable resources by entering values in the group names column.
  - For movable resources a group name would most likely represent a workload group that can be used in a 'move group' command.
  - For non-movable resources the group names serve no useful purpose in the current version of SSMGA. In a future release, the group names will be used for global group monitoring similar to the current SSM Group Manager application.
  - Any group names specified are implicitly global in scope and must have a unique significance within the SSMplex.
- Define cross-system resource prerequisites by updating the prereq column with the full cross-system name of the resource:

  `system.subsys.table.name`

  For movable resource workloads that must always run on the same system, the following user designed prereq is provided in SSMGA to indicate system affinity with the base resource of the workload group:

```
*SYS.AFF.table.name
```

This prereq prevents moved resources from starting on the new system until all the resources in the group have shutdown and moved to the new system.

- Populate the alternate system list column with the desired system names specified in the order they are to be selected.
- You can update any combination of the PRIMARY_SYSTEM, SSM#MOVMOD and the SSM#SYSLST column for a single resource by using the SSMGAMRS routine contained in your SYS1.CCLXEXEC library. It is particularly useful for modifying multiple copies of a movable resource without having to modify each copy manually. The syntax of SSMGAMRS, together with all possible keywords, is as follows:

```
SSMGAMRS {prisys.}table.resource PRISYS() MOVMOD() SYSLST() SYSADD() SYSDEL()
SSMPLEX() SYSTEM()
```

{prisys.}table.resource

Is required and refers to an individual resource that is to be modified. Specify *table.resource* with, or without the primary system portion of the name. At least one keyword of *PRISYS*, *MOVMOD* or *SYS***.*

Example:

SYSLST or SYSADD and/or SYSDEL must be specified alongside this resource name.

PRISYS()

Can be used to change the primary system name of the resource. It must be in the format *system.subsys*. If used in combination with SSMPLEX keyword, every copy of the specified resource will have its PRIMARY_SYSTEM column updated to the value you specified here. If a new Primary system is specified, and it appears in the current system list of the resource you are modifying, then it is removed from the system list (unless a new complete SYSLST is also specified).

MOVMOD()

Can be used to specify a new move mode for the resource.

SYSLST()

Can be used to specify a new complete system list to go in the SSM#SYSLST column for the resource. It will replace any existing values in the SSM#SYSLST column if used.

SYSADD()

Can be used to add a single new system to the existing SSM#SYSLST column for the resource. It consists of two comma-separated operands: the system name in the format *system.subsys* and the position in SSM#SYSLST that the new system name will be placed. For example, SYSADD(SYSA.OPSS,1) will add system SYSA.OPSS to the front of the system list in column SSM#SYSLST for the resource specified.

SYSDEL()

Can be used with or without SYSADD to remove system names. The value *ALL indicates an empty SYSLST.

SYSTEM()

Accepts a list of system names in the format system.subsys. Only copies of the resource on those systems in the list are updated.

SSMPLEX()

Accepts the name of an SSMPLEX that the resource is part of. Only copies of the resource on that SSMPLEX will be modified. If SSMPLEX is not specified then the resource on all MSF connected systems in the current system's SSMPLEXNAME are updated. If SSMPLEX(NONE) is specified , then only the system names in the SSM#SYSLST are updated together with the primary system.

Step 8: Add SSM Action Table Entries

SSMGA requires that the following actions be added to the SSM resource action tables.

```
ACTION_ ACTION_   ACTION_

PROCESS CURRENT   MODE       ACTION_TEXT
MSUBREQ         …            RULE("SSMGAMSR &SSM!TABLE &SSM!NAME")
SELECT          …          … PROCESS("XPREREQ,XSUBREQ,MSUBREQ ")
SELECT          … INACTIVE … PROCESS("NOSUBREQ,NOPREREQ,NOACTION")
SELECT  UNKNOWN … INACTIVE …
XSUBREQ         …          … RULE("SSMGAPRX &SSM!PROCESS &SSM!XRESNAME")
XPREREQ         …          … RULE("SSMGAPRX &SSM!PROCESS
                                    &SSM!PREREQ PRERES(&SSM!RESNAME)")
```

Note: (_) denotes one or more empty columns in the table entry.

Step 9: Replicate Parameters, RDF Tables, and Rules

Each eligible system (the primary system and all alternate systems) for each movable resource must have an identical resource table definition, an identical associated action table, and identical AOF rules (if not using shared AOF rule sets) for that movable resource. Required SSMGA parameters must be set on each system that will participate in the global SSMplex.

Note: Before executing this step, it is strongly recommended that you deactivate SSM processing on each system that will participate in the global SSMplex by setting the STATEMAN parameter as follows:

```
STATEMAN=INACTIVE
```

To replicate parameters, RDF tables, and rules

1. On each eligible system, replicate resource tables containing movable resources by using the copy function of either the RDF Table Editor (OPSVIEW 2.6) or the SSM Resource Editor (OPSVIEW 4.11.R).
2. On each eligible system, replicate action tables associated with resource tables containing movable resources by using the copy function of either the RDF Table Editor (OPSVIEW 2.6) or the SSM Action Editor (OPSVIEW 4.11.A).
3. If not using shared AOF rule sets, on each eligible system, replicate and auto-enable any AOF rules (such as start, stop, state detection, and so on) for a movable resource.
4. If not using a shared OPSPA00 initialization OPS/REXX program, update the OPSPA00 initialization OPS/REXX program with the required SSMGA parameters on each system that will participate in the global SSMplex.

Step 10: Activate SSM Processing and Verify the Setup

After making all changes to an existing SSM configuration, you need to activate SSM processing and then verify that the setup is complete.

To activate SSM processing and verify the setup

1. Set the parameter STATEMAN as follows:

   ```
   STATEMAN=ACTIVE
   ```

   The SSM processing is activated.

   Note: SSMGA is not activated until the SSMSUBRULE parameter is set to SSMGAGBL. See Step 4 above, and section Using the SSM Subtask below.

2. Verify the setup by issuing the VERSYS command from the OPSVIEW 4.11.G panel.

   The SSMGA local setup is validated.

Using the SSM Subtask

You control the starting and stopping of the SSM subtask using the following SSMSUBRULE parameter values:

- A value other than 'NONE' starts the SSM subtask
- Setting the value to 'NONE' stops the SSM subtask

Note: The value of the SSMSUBRULE parameter must be an enabled AOF request rule name in order for any productive work to be performed.

The SSM subtask is posted for an event by any of the following conditions:

- An explicit post by the OPSSMTBL command with the keyword operand POST(SSMSUB).
- Anytime an MSF connection with the same SSMPLEXNAME value changes to INACTIVE status.
- Anytime an MSF connection's SSMPLEXNAME, SSMACTIVEGLOBAL, or SSMPRIORITY value changes.
- Anytime an external RDF table modification SQL statement changes a table whose name matches the table name prefix specified in the SSMSUBPREFIX parameter. Table modifications that originate from the local SSM main task and the SSM subtask do *not* cause a post event.

Although the new SSMGA feature will use this subtask for transmitting status information from system to system, the use of the SSM subtask is not limited to SSMGA. You can use this subtask to perform continuous auxiliary automation in support of SSM events.

Some possible uses for the subtask are as follows:

- Execute a designated AOF request rule each time it is posted
- Handle repetitive SSM processing that involves long execution times that could impede the performance of the primary SSM task
- Implement an alternate automation procedure such as MSF connection management

Messages for Special Events

SSMGA issues the following WTOR messages when human intervention is required:

- SSMGA01O
- SSMGA02O
- SSMGA03O
- SSMGA04O

You can develop automated responses to these WTOR messages using AOF message rules. Pay careful attention to the consequences of answering these messages incorrectly.

Note: For detailed descriptions of these messages, see the *Message Reference*.

SSMGA Status Command

An CA OPS/MVS modify command provides the return status for OPS/MVS, MSF, SSM, and SSMGA. This command determines the SSMGA system status within a sysplex when MSF communication has been lost. The z/OS sysplex ROUTE command can be used to issue this modify command on the target system within the same sysplex. Examination of the z/OS or the OPS/MVS messages returned can determine if the loss of communication warrants the recovery or movement of SSMGA resources. The SSMGA automatic WTOR reply Rexx program, SSMGARPL, uses this command.

Command Syntax

```
MODIFY OPSx,STATUS(SSMGA)
```

where x = the fourth character of your CA-OPS/MVS subsystem name.

Returns message

```
OPx3269I SSMGA: OPS=var1 MSF=var2 SSM=var3 SSMPLEX=var4 GBL=var5 PRI=var6
```

var1

CA OPS/MVS product status

var2

MSF component status

var3

Stateman mode (Same as OPSINFO)

var4

SSMPLEXNAME (name/NONE). Used in SSMGA.

var5

SSMACTIVEGLOBAL value (Y/N). Used in SSMGA.

var6

SSMPRIORITY (0-1999). Used in SSMGA.

Example

```
F OPSK,STATUS(SSMGA)
OPK3269I SSMGA: OPS=ACTIVE MSF=ACTIVE SSM=ACTIVE SSMPLEX=QSSMPLEX GBL=Y PRI=300
```

Verification and Diagnostic Commands

Several ISPF primary commands for detecting SSMGA configuration and runtime errors are available in the OPSVIEW 4.11.G SSMGA resource monitor. These commands detect the most common errors in SSMGA and in a few cases repair them automatically. When new SSMGA tables are added to the configuration, to insure a valid configuration all of these commands should be executed in order.

VERSYS command (Rexx program: SSMGAPH1):

- Validates the local system SSMGA required parameter settings
- Checks for MSF connections to an SSMGA global system
- Checks for required tables and their column structures
- Checks SSMGA resource column values for syntax, contents, and consistency
- Verifies that required rules are enabled
- Checks action table columns for the SSMGA required actions
- For some parameter and table errors the proper SSMGA value is set

VERGBL command (Rexx program: SSMGAPH2):

- Verifies that all SSM resource table rows on accessible systems in the SSMplex have corresponding resource rows in the local LST and in the GST
- Detects rows in the LST and GST that do not have corresponding SSM resource rows

VERMOV (Rexx program: SSMGAPH3:

- Diagnoses problems with movable resources
- Identifies any outstanding SSMGA WTORs that require a response
- Verifies that a specific movable resource has the same primary system name on all systems
- Verifies that movable resources exist on systems that are consistent with their primary and alternate system definitions
- Identifies movable resources that are not assigned to a system or are waiting for a particular system that is not active
- Evaluates the current and desired system in light of the action mode and move mode for consistency on all systems

Chapter 9: Using Group Manager

This section contains the following topics:

Monitor Groups of Managed Resources

Use the Group Manager feature of CA OPS/MVS to group system resources under the control of the System State Manager and to monitor the status of resource groups.

To help you find resources that are not in their correct state among the hundreds or even thousands of system resources that the System State Manager may be monitoring at your data center, the System State Manager offers a Group Manager feature. Using the Group Manager and its tables and ISPF panels, you can:

- Categorize resources that the System State Manager monitors into groups.
- Monitor the status of groups from all systems connected through MSF to CA OPS/MVS on the local system, or monitor only a subset of those groups.
- Display a list of the resources belonging to each group.
- For each resource, change the desired state, the mode in which System State Manager monitors the resource, and whether the status of the resource contributes to the status of the group to which it belongs.

To use Group Manager

1. Set the STATEGROUPMAN parameter to YES
2. Set the System State Manager to active.

For more detailed information on the STATEGROUPMAN parameter, see the *Parameter Reference*.

Tables Used by the Group Manager

To keep track of the status of groups and resources, the Group Manager uses four relational tables:

- The group membership table SSM_GROUP_MEMBERS defines groups of resources. You can edit this table using the CA OPS/MVS table editor.
- The status definitions table SSM_GROUP_STDEF defines the statuses groups can have and how those statuses display on Group Manager panels. You can edit this table using the CA OPS/MVS table editor.
- The status selection table SSM_GROUP_STSEL is used to determine the current status of resources in each group. You can edit this table using the CA OPS/MVS table editor.
- The current status table provides the data that is to be displayed by the Group Manager monitor. The Group Manager creates this table at System State Manager startup and updates it for each monitor display. Using the current and desired states of each resource being monitored, the System State Manager assigns a status and priority in this table for each group in which the resource participates. This table is used to determine the display status of each group being monitored.

WARNING! Do not edit the current status table; doing so can seriously disrupt Group Manager and System State Manager operation.

Define Groups and Assign Resources to Them

An individual resource can belong to more than one group, and each group can contain more than one resource. The group membership relational table SSM_GROUP_MEMBERS defines which resources belong to which groups.

The SSM_GROUP_MEMBERS has the following format:

M_TABLE

The name of the table that defines the resource to System State Manager

Type: CHAR(18)

Primary Key: Yes

M_RESOURCE

The name of the resource

Type:CHAR(18)

Primary Key: Yes

M_GROUP

The name of the group to which the resource belongs

Type: CHAR(18)

Primary Key: Yes

The group membership table on a given system is automatically created by Group Manager and initially consists of data extracted from the SSM_MANAGED_TBLS table on that system. Each table in SSM_MANAGED_TBLS is defined as a group, and all resources defined in a table become members of the same group. To edit a group membership table, select the following option from the Group Manager Main Menu (OPSVIEW option 4.11.5) and specify an MSF system ID (defaults to the local system):

```
_ Specify group membership on system ===> ________
```

Define Statuses for Your Groups

Each group you create can be in a variety of statuses.

To define most of these statuses, edit the status definitions table SSM_GROUP_STDEF.

The following describes the SSM_GROUP_STDEF table columns:

D_STATUS

The name of the status.

Type: CHAR(18)

Primary Key: Yes

D_GROUP

The name of the group to which the status belongs.

Type: CHAR(18)

Primary Key: Yes

D_PRIORITY

The priority of this status, a value from 1 (highest priority) to 231 -1 (lowest priority).

Type: INTEGER

Primary Key: No

D_COLOR

The color to use when displaying the status text. Possible colors are:

- Blue
- Green
- Pink
- Red
- Turquoise
- Yellow
- White

You can abbreviate the color names, using only the first character of each name.

Type: CHAR(9)

Primary Key: No

D_HIGHLIGHT

The type of highlighting to be used (if your terminal allows highlighting) when displaying the status text. You may specify only one of the following possible types of highlighting:

- None (abbreviated as NON)
- Normal (abbreviated as NOR)
- Blinking (abbreviated as B)
- Reverse (abbreviated as R)
- Underscore (abbreviated as U)

Type: CHAR(10)

Primary Key: No

D_TEXT

The text that the Group Manager displays when a resource or a group is in this status.

Type: CHAR(120)

Primary Key: No

Define Status Names

You can define your own names for the statuses of resources. Define names that point to the condition that the statuses represent. For instance, you could define WTNG_MOUNT as the name for the status of a tape drive that is being mounted.

CA OPS/MVS uses three default status names:

- CUR_EQ_DES (the current state of the resource equals its desired state)
- CUR_NE_DES (current state of the resource does not match its desired state)
- DEF_ERROR (status name not defined in the SSM_GROUP_STDEF table is referenced in another table called SSM_GROUP_STSEL)

Unless you define them differently in the SSM_GROUP_STDEF table, these three statuses have the following characteristics:

CUR_EQ_DES

D_Group Column: No group

D_PRIORITY Column: 231 - 1 (lowest possible priority)

D_COLOR Column: GREEN

D_HIGHLIGHT Column: No highlighting

D_TEXT Column: Resource &RN State &CS (&RN is a variable containing the name of the resource)

CUR_NE_DES

D_Group Column: No group

D_PRIORITY Column: 1 (highest priority)

D_COLOR Column: RED

D_HIGHLIGHT Column: No highlighting

D_TEXT Column: Resource &RN State &CS (&CS is a variable containing the current state of the resource)

DEF_ERROR

D_Group Column: No group

D_PRIORITY Column: 1

D_COLOR Column: RED

D_HIGHLIGHT Column: UNDER (underscore)

D_TEXT Column: Status status Not Defined

Example: SSM_GROUP_STDEF table

The following is an example of the SSM_GROUP_STDEF table:

```
D_STATUS           D_GROUP     D_PRIORITY  D_COLOR   D_HIGHLIGHT D_TEXT
************************** TOP OF DATA ********************************* TOP OF DATA ******************************
MATCH                      200         GREEN     NULL      DESIRED_STATE MATCHES CURRENT_STATE
MISMATCH                   1           RED       B         DESIRED_STATE DOES NOT MATCH CURRENT_STATE
TRANSIENT                  100         YELLOW    NULL      RESOUSCES IN TRANSIENT STATUS
UNKNOWN_ERR                25          TURQUOISE R          MISMATCH IN DESIRED_STATE AND CURRENT_STATE. CHECK SSMSTATE
UP_IPL                     50          PINK      U         &RN IS UP
```

Associate a Status with a Group or Groups

To have a status apply to a particular group, specify the group name in the D_GROUP column. If you specify no group name, the status applies to all groups.

Set the Priority of a Status

Assigning a priority to a status allows the Group Manager panels to call the attention of the system operator to statuses reflecting resource availability problems. For example, the status name CUR_NE_DES defaults to the highest priority, 1, because it indicates that the current state of a group does not match the desired state.

Note: Differences between these states usually point to a resource problem that the system operator needs to know about.

Conversely, the status CUR_EQ_DES indicates that the current and desired states match, which usually means that the resources in a group are operating without problems. Therefore, the CUR_EQ_DES status has the lowest possible priority.

The Group Manager requires you to specify a priority for each status you define. If two resources in the same group have statuses with the same priority, the resource that had that priority first determines the group status.

The Group Manager takes the status of a group from the resource that has the highest-priority status. For example, suppose that in a group of 10 DASDs, one has the status CUR_NE_DES but the other nine DASDs have low-priority statuses. The Group Manager assigns the high-priority CUR_NE_DES status to the entire group. This allows the operator viewing Group Manager panels to detect a problem with this group of DASDs, then use the Group Members panel of the Group Manager to identify which DASD has the problem.

Use Substitution Parameters in Status Text

As you add status definitions to the SSM_GROUP_STDEF table, you can use substitution parameters in defining the text that goes into the D_TEXT column, that is, the text that describes a status on Group Manager panels. These parameters allow you to determine which status data is most relevant. For example, by putting parameters referencing the resource name and current state into the status text, you allow an operator viewing group display of the Group Manager to see the current state of the resource that is causing a problem.

All substitution parameters begin with an ampersand (&). You can place a period after a parameter when you want to concatenate it in front of normal text.

Parameter	Substitution Value
&CS	Current state of the resource
&DS	Desired state of the resource
&GN	Name of the group to which the resource belongs
&GO	Order that the Group Manager uses to sort resources by priority

Parameter	Substitution Value
&PF	Whether the resource has missing prerequisite resources
&RN	Resource name
&RT	Resource table
&RY	Resource type
&SC	Color in which the status text appears
&SD	The date when this status was assigned to the resource or group
&SH	Highlighting used to display status text
&SM	System State Manager processing mode
&SN	Status name
&SP	Status priority
&ST	The time when this status was assigned to the resource or group

How Group Manager Assigns Statuses to Resources and Groups

The current and desired state of a resource is used by Group Manager.

Group Manager initiates the following process:

- Assigns a status to each resource for each group.
- Uses the status of each resource in a group to assign a status to each group.
- Stores current status information for each resource for each group in the current status table.

  The current status table is built dynamically by Group Manager and should not be altered by the user.
- Builds the current status table using the status selection table built by the user.

Important! Do not edit the current status table; doing so can seriously disrupt Group Manager and System State Manager operation.

Status Selection Table

The status selection table SSM_GROUP_STSEL contains the data required to associate a status to each combination of resource states for each group.

The structure of the columns status selection table is as follows:

S_GROUP

Specifies the group name

Type: CHAR (18)

Primary Key: Yes

S_CURRENT

Current state of a resource

Type: CHAR (8)

Primary Key: Yes

S_DESIRED

Desired state of a resource

Type: CHAR (8)

Primary Key: Yes

S_TABLE

Resource table name

Type: CHAR (18)

Primary Key: Yes

S_TYPE

Resource name or resource type (if the type column is present)

Type: CHAR (18)

Primary Key: Yes

S_STATUS

Name of the status to be assigned; must be defined in SSM_GROUP_STDEF

Type: CHAR (18)

Primary Key: No

The status selection assignment algorithm is similar to that of the System State Manager action table search. In a matching group name and current state, the most specific match of the remaining key fields in hierarchical sequence is selected. For instance, a match on desired state outweighs a row where desired state is null but resource table name and resource name match. Once the status name is assigned, the status definition table is accessed to obtain priority and display data.

If no status selection table rows are defined for the group name, default status rows without a group name may also be defined for each current state value and subsequent portions of the table key. If the default rows are not in the table, the status CUR_EQ_DES or CUR_NE_DES is assigned as appropriate.

The following is an example of the SSM_GROUP_STSEL table:

```
S_GROUP       S_CURRENT S_DESIRED S_TABLE            S_TYPE      S_STATUS
********************** TOP OF DATA ****************************************
              DOWN      DOWN                                     MATCH
              DOWN      UP                                       MISMATCH
              STARTING  UP                                       TRANSIENT
              STOPPING  DOWN                                     TRANSIENT
              UNKNOWN   DOWN                                     UNKNOWN_ERR
              UNKNOWN   UNKNOWN                                  MATCH
              UNKNOWN   UP                                       UNKNOWN_ERR
              UP        DOWN                                     MISMATCH
              UP        IPL                                      UP_IPL
              UP        UP                                       MATCH
```

Use the Group Manager Displays

Group Manger displays let you view resource and resource group information.

To use Group Manager Displays

1. Select System State Manager from the OPSVIEW primary panel.
2. Choose option 5 from the Ssytem State Manager Functions panel to view the resource and resource group information displayed on Group Manager panels.

   The Main Menu of the Group Manager appears.

```
Group Manager --------------------- Main Menu ---------------------------------
COMMAND ===>

------------------------------ Individual Resources --------------------------
  _ Specify which systems you want to monitor
  _ Specify which groups you want to monitor and their position on the display

------------------------------ Individual Parameters -------------------------
    Refresh the monitor display every 15_ seconds.

------------------------------- Global Resources -----------------------------
  _ Specify group membership on system ===> ________
  _ Specify group status selection on system ===> ________
  _ Specify status definitions (colors, text, etc.) on system ===> ________
  _ Synchronize with the StateMan table definitions on system ===> ________


To select any option, type S beside it and press ENTER.
Press ENTER with no options selected to enter the monitor display
```

3. Choose one of the options from the Main Menu by typing an **S** beside it and pressing the Enter key.

   You receive one of the following panels:

If you select...	The following happens...
Specify which systems you want to monitor	You see the System Selection List panel
Specify which groups you want to monitor	You see the Group Selection/Ordering panel
Enter, that is, if you press Enter without selecting any Main Menu options	You see the Group Display panel
Specify group membership	The CA OPS/MVS table editor is entered for table SSM_GROUP_MEMBERS
Specify group status selection	The CA OPS/MVS table editor is entered for table SSM_GROUP_STSEL

If you select...	The following happens...
Specify status definitions	The CA OPS/MVS table editor is entered for table SSM_GROUP_STDEF
Synchronize with Stateman table definitions	A message is displayed on the main menu panel indicating that System State Manager has been told to rebuild the current status table

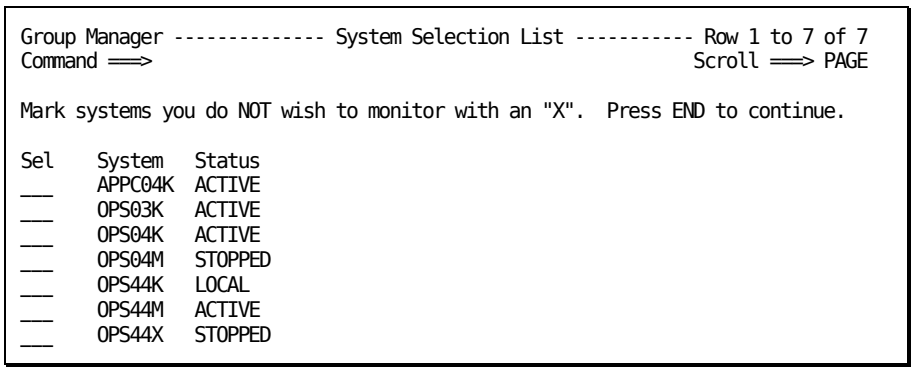
Exclude Systems from Resource Monitoring

Some systems do not need monitoring and should be excluded from the list of monitored systems.

To exclude some systems from being monitored

1. In the System Selection List panel, type an X in the Sel column beside the systems to be excluded. If a system is stopped, you need not exclude it.

   The Group Manager stores the list of excluded systems, and any changes to that list, in your ISPF profile. Once excluded from monitoring, a system stays excluded until you include it (by removing the X beside it).



2. When you have indicated which systems the System State Manager should exclude from monitoring, press the End key.

Choose Resource Groups to Monitor

Use the following Group Selection/Ordering panel to specify:

- Which groups of resources System State Manager should not monitor.
- The order in which monitored groups will appear on the Group Monitor Display panel. Specifying a display order for groups ensures that status information for a group always appears on the same line of the screen, allowing on operator to determine the status of a group at a quick glance from the position and color of the line.

```
Group Manager ---------- Group Selection/Ordering List ----- Row 1 to 10 of 10
Command ===>                                                  Scroll ===> PAGE

Mark groups you do NOT wish to monitor with an "X" or place a number next to a
group to position it on the group display.  Press END to continue.

SEL     SYSTEM   GROUP             STATUS
___     OPS03K   CICS1             ACTIVE
___     OPS03K   CICS2             ACTIVE
___     OPS04K   NETSPY            ACTIVE
___     OPS04K   ASTEX             ACTIVE
___     OPS04K   SYSVIEW           ACTIVE
___     OPS44K   BUNDL             LOCAL
___     OPS44K   NVISION           LOCAL
___     OPS44K   ASTEX             DELETED
___     OPS44K   SYSVIEW           DELETED
___     OPS44K   CICS3             LOCAL
```

To choose resource groups to monitor

1. Exclude a group from System State Manager monitoring by typing an X beside it in the Sel column.
2. Specify where the status information of a group appears in the displayed group list by entering a number instead of an X.

   The position of this information is relative. For instance, if you assign the numbers 1, 9, and 25 to three groups, the group marked with number 1 appears on the first line the group indicated with 9 appears on the second line, and the group indicated with 25 appears on the third line.

   As you add groups, the Group Manager automatically adds them to the Group Ordering/Selection List panel, listing them ahead of the groups for which you specified positions. Groups excluded from monitoring appear at the bottom of the list.

   Note: DELETED status means that although you have assigned a position to this group, the administrator of the system where this group resides has deleted that group. STOPPED status indicates that you have assigned a position to this group but the MSF link to the system of that group is currently down.
3. When you have finished specifying which groups should be monitored and the screen positions of those groups, press the End key.

The Group Display panel displays.

View the Status of Groups

The following Group Display panel lists the status of all resource groups that you have asked to be displayed.

To view the status of groups

1. Press Enter on the Main Menu of the Group Manager without selecting any options.

   The following the Group Display panel displays:

```
Group Manager ------------------------ Group Display -------------------------
Command ===>                                                 Scroll ===> PAGE
CMDS: A R S (Type MONitor on the command line for continuous monitoring.)
     SYSTEM   GROUP-NAME         STATUS
__   OPS04K   NETSPY             MISMATCH IN DESIRED_STATE AND CURRENT_STATE
__   OPS04K   ASTEX              MISMATCH IN DESIRED_STATE AND CURRENT_STATE
__   APPC44K  SYSVIEW            DESIRED_STATE MATCHES CURRENT_STATE
__   OPS44K   NVISION            DESIRED_STATE MATCHES CURRENT_STATE
__   OPS44K   CICS3              GROUP HAS NO MEMBERS
```

   The priorities you assign to groups on the Group Selection/Ordering List panel determine the order in which groups appear on the Group Display panel. Groups not excluded or that have no assigned position appear first, sorted in order of the priority of their status. Groups assigned the same relative position are sorted by status priority.

2. Enter the following three line commands on the Group Display panel to view the status of the groups:

   A

   Causes the Group Manager to display its Group Members panel, which lists all the resources in the selected group

   R

   Goes directly to the Resource Details Display panel for the resource that is determining the status of the group

   S

   Causes the Group Manager to display only the resources in the selected group whose current states do not match their desired states

Note: When the Group Display panel is in automatic monitoring mode, you must press the PA1 key before entering these line commands.

View the Status of Group Members

The Group Members panel, shown here, displays the status of resources in the currently selected group:

```
Group Manager: ------------ Members of group GROUP1 -------- System: S008
COMMAND ===>                                              SCROLL ===> PAGE
 CMDS: Q R S (Type MONitor on the command line for continuous monitoring
 updates)
        Table-Name         Resource-Name     Status
___     STC_TABLE          RESOURCE4         RESOURCE4 IN STC_TABLE IS DOWN
___     T2                 RESOURCE2         RESOURCE2 IS IN STATE5
___     T2                 RESOURCE3         RESOURCE3 IS IN STATE3
___     T2                 RESOURCE1         RESOURCE1 IS IN STATE2
___     T1                 RESOURCE1         OK
___     T1                 RESOURCE3         RESOURCE3 IS IN STATE6
___     T3                 RESOURCE1         RESOURCE1 IS IN STATE4
```

The Group Members panel has three line commands:

Q

Toggles the resource in and out of Group Manager quiesced mode. While a resource is in quiesced mode, its status cannot be used as the status of a group. So, if a quiesced resource is the only resource in a group operating abnormally, then the status of the group is OK. To remind you that a resource is quiesced, the letter Q appears immediately after the CMDS column of the Group Display and Group Members panels. Quiesce mode is useful when a resource is in an abnormal status but the operator can do nothing about it, for instance when a piece of hardware has a problem that will make it inoperative until someone replaces a part.

R

Included for compatibility with the Group Display panel, the R command on the Group Members panel has the same function as the S command.

S

Causes the Group Manager to display the Resource Details Display panel for only the selected resource.

Note: When the Group Members panel is in automatic monitoring mode, you must press the PA1 key before entering these line commands.

Automatically Monitor Groups or Resources

You can have Group Manager automatically monitor the status of groups or resources.

To monitor groups or resources automatically

1. Type MONITOR (or MON) primary command in either the Group Display panel or the Group Members panel.

   The MONITOR (or MON) primary command checks the status of all resources every *nn* seconds and refreshes the display. The *nn* value is the number of seconds between refreshes that you specify on the Main Menu of the Group Manager. During monitoring, you have an up-to-date display of the status of the groups at all times without having to enter any keystrokes.

2. To turn monitoring off, press either the ATTN key or the PA1 key.

   Automatic monitoring is turned off.

View Detailed Resource Information

The following Details Display panel displays status information for a specific resource:

```
Group Manager: Resource Details Display--------------------------------
 COMMAND ===>

Resource information:
   Group    ===> CICS
   Table    ===> GRG_STCTBL
   Type     ===> CICS
   Resource ===> CICSA

Group status information:
   Name     ===> CUR_EQ_DES
   Priority ===> 258
   Quiesced ===> NO
   Text     ===> CICSA IN STATUS CUR_EQ_DES AT 00:45:58 ON 05/30/04

SSM state information:
   Current  ===> UNKNOWN (To set current state to UNKNOWN type S here ===>  )
   Desired  ===> UNKNOWN
   Mode     ===> ________
   Missing  ===>
   Prereqs
```

From this panel you can change the resource's:

- System State Manager desired state
- System State Manager processing mode
- Group Manager quiesced mode

To change any of these values, enter new values over those shown on the panel. The current state can only be changed to the unknown state by entering an S in the input field. System State Manager re-determines the correct current state as prescribed by the unknown action in the action table.

Exit from Group Manager Panels

To exit from any Group Manager panel and return to the previous panel, issue the END command from the command line.

Chapter 10: Using Schedule Manager

This section contains the following topics:

Reasons to Use the Schedule Manager

Using the CA OPS/MVS Schedule Manager facility, you can schedule the desired state of one or more resources by date, by day of the week, or by time of day.

Being able to schedule the desired state of resources enables you to automatically configure them at any time, including system IPL time, when batch job schedules may not be active. The only requirement is that CA OPS/MVS be active. You can define many alternate configurations of system resources into one or more Schedule Manager schedules. Some examples of alternate system configurations are:

- Shift changes
- Company holidays
- Weekend test time
- Disaster recovery mode

The benefits of using the Schedule Manager to manage the state of system resources are:

- No time-of-day (TOD) rules are used. This eliminates the need to code sophisticated catch-up logic.
- Sophisticated rules or REXX routines no longer need to be created to verify what is running.
- Schedules are created and maintained through ISPF.
- Recovery from unplanned system outages is automatic. The Schedule Manager configures your system resources in the proper state no matter what date, day of the week, or time of day the unplanned outage occurs. You do not need to manually verify what should be active and what should be inactive.

  For example, if certain printers defined to System State Manager need to be available only during regular working hours (such as Monday through Friday between 7:00 a.m. and 8:00 p.m.), then you can use the Schedule Manager to change the desired state of those printers to DOWN between 8:00 p.m. and 7:00 a.m. on weekdays and all day on Saturday and Sunday. You can also have the Schedule Manager automatically change the desired state of the printers back to UP on Monday through Friday morning at 7:00 a.m.

  It is possible to manually override Schedule Manager control of specific resources while maintaining usual scheduled control over remaining resources. Using the example above, if a printer needed to be UP for an hour at noon on a Saturday, then various mechanisms in the Schedule Manager exist to allow the printer to be brought UP without having to unload the active schedule.

Note: The Schedule Manager is *not* a batch job scheduler.

What You Can Do With Schedule Manager

You can define one or more schedules to the Schedule Manager; however, only one schedule is active at a time. Note that it is possible to merge multiple schedules into one schedule. In each schedule, define one or more periods of time by date, by day of the week, and by time of day. To schedule the desired state of a System State Manager resource, link both it and its desired state (ON or OFF) to a period of time in a schedule. After you tell the Schedule Manager what schedule to activate (or load), the Schedule Manager determines what periods in the schedule are active and sets the scheduled desired state of any resources linked to those periods. This process is called reset processing. The time of day when a scheduled period starts or ends is a time of possible change in desired states, and the Schedule Manager automatically performs reset processing at those times. To manually request a Schedule Manager reset, use the Schedule Manager SSM@OVER sample command rule or use the Schedule Manager API.

Note: The Schedule Manager stores period and schedule definition data in the database of the Relational Data Framework (RDF). It uses an internal schedule called ACTIVE to set the desired state of resources under System State Manager control. You cannot edit the ACTIVE schedule, but you can create a new schedule or edit an existing schedule, and then load it as the ACTIVE schedule.

The remainder of this chapter provides information on how to do the following:

- Add, copy, delete, edit, print, or rename a schedule
- Add, delete, edit, or rename a period in a schedule
- Link or unlink resources and periods
- Synchronize a schedule with System State Manager resource table updates
- Merge the information from two schedules into a single schedule
- Load a schedule to make it the ACTIVE schedule
- View or print schedule period overlaps and conflicts
- View or print the schedule state of system resources
- Override the ACTIVE schedule
- Use the Schedule Manager application program interface (API)

Note: Most Schedule Manager functions can operate using schedule data on any MSF-connected system.

Definition of Terms

The description of the Schedule Manager in this document and the Schedule Manager panels use the following set of special terms:

Schedule

A complete, self-contained set of scheduling information, including:

- A list of time periods included in this schedule. See the description of period below.
- Information about system resources and their respective states during a specified time period.

Resource

A component of your system, such as a DASD controller, a printer, or an application program. The Schedule Manager schedules desired state changes only to resources defined in one of the relational tables included in the SSM_MANAGED_TBLS table of the System State Manager.

Period

The dates and times governing when System State Manager changes the desired state of one or more resources. A period has start and end times specified in hours and minutes and either a cyclic or a fixed start date and end date:

- A **cyclic date** consists of one or more days of the week (Sunday, Monday, and so on). If you use a cyclic date, then the period takes effect **every week** on the same day or days.
- A **fixed date** consists of specific days of specific months. If you specify one fixed date, then the period takes effect only on the specified day (for example, March 27). If you specify two fixed dates, then the period begins on the first date and ends on the second date. For example, if you specify 11/11-12/12, the period begins on November 11 and continues until December 12. You can optionally specify the year of the first date. If the year of the first date is specified, then the period is active only once beginning on the first date in the specified year. If the year of the first date is not specified, then the period is active every year beginning on the first date (unless it is February 29, which is active in leap years only). The year of the second date is assumed from the first date, whether a year has been specified on the first date. A fixed date period cannot span more than 365 days.

A single period can change the desired state of many resources, and more than one period can control the desired state of a single resource.

Period set

All periods that apply to a specific date and time.

Link

A link is the association between a resource and a period.

Link item

A link item is either the resource or the period that is part of a link.

Link item column

The column on Schedule Manager panels containing the link item. For example, if the link item is a resource, then the resource column is the link item column.

Link group

The group of system resources or periods that is associated with the link item. For example, if the link item is a resource, then the link group is all periods that change the desired state of that resource.

Link group column

The column on Schedule Manager panels that contains the link group. For instance, if the link item is a period, then the link group column is the resources column.

Conflict

A Schedule Manager conflict occurs when two periods specify opposite desired states for the same resource at some point in time.

Conflict group

The periods specifying conflicting desired states for a resource.

Overlap

The situation that occurs when a new period begins before the previous period ends. Overlapping periods may or may not produce a conflict.

Commands column

The column on Schedule Manager panels where you issue line commands. When the commands column resides in the period column (its default location), you can issue line commands against periods. To issue line commands against resources, you need to move the commands into the resource column. PF11 moves the commands column to the resource (right) side of the screen; PF10 moves the commands column to the period (left) side of the screen.

Perform Schedule Manager Tasks

The Schedule Manager lets you control the states of system resources over specific time periods, both on the system you are working on, and on other systems that are running the Schedule Manager.

To access Schedule Manager and perform tasks

1. From the CA OPS/MVS main OPSVIEW panel, select option 4.11.4.

   The following Schedule Manager Primary Panel displays:

```
Schedule Manager --------------- Primary Panel --------------------------------

COMMAND ===>
       C - Copy schedule                D - Delete a schedule
       E - Edit a schedule              F - Free a schedule
       I - Insert a new schedule        L - Load schedule (make it ACTIVE)
       M - Merge schedules              R - Rename schedule
       Y - Synchronize schedule     blank - Display schedule list

SPECIFY SCHEDULE:
   NAME    ===>                        (If option C,D,E,F,L,M,R or Y selected)
   NEWNAME ===>                        (If option C,I,M or R selected)

CONFIRM DELETES: YES (Enter YES to require delete confirmation)

NOTE: To use a schedule on another system specify the name as system>schedule
      Specify ? as the system name to get a list of all systems.
```

2. From this panel, choose a schedule maintenance task by issuing one of the listed commands from the command line and specifying the name of the schedule you want in the NAME field (and if required, the name of a new schedule in the NEWNAME field).
3. On any Schedule Manager panel, press Enter

   The command or action you have specified executes.

For more information about the main Schedule Manager panel, or any OPSVIEW panel, see the OPSVIEW *User Guide*.

Select a Schedule

The Schedule Manager can maintain schedules defined on the system you are working on, as well as schedules from other systems running CA OPS/MVS and the Schedule Manager.

Schedule names are in the format of *sysname*>*schedname*, where *sysname* is the name of the system and *schedname* is the name of the schedule. For example, SYSA>SCHEDULE1.

Schedule lists can be displayed in a variety of formats.

To select a schedule or schedule list, use the following chart. This chart describes what you need to enter to view a specific schedule or schedule list:

To select...	Enter this value in the NAME field...
One of the schedules defined on the current system	The name of that schedule in the format *schedname* or *sysname*>*schedname*, where *sysname* is the name of the current system
A schedule from a list of schedules on the current system	An asterisk (*), or leave the NAME field blank
A specific schedule on another system	The name of the system followed by the greater-than symbol (>) and the schedule name, as shown in the following example: *sysname*>*schedname*
A list of all systems having defined schedules	?
A list of all schedules on all systems	>
A list of all schedules named *schedname* on all systems	>*schedname*
A list of all schedules on the specified system	*sysname*>
All schedules beginning with the characters SCHED on the current system	SCHED*
All schedules beginning with the characters SCHED on all systems	>SCHED*
All schedules beginning with the characters SCHED on a specific system	*sysname*>SCHED*, where *sysname* is the name of the specific system

Insert a New Schedule

To create a new schedule using Schedule Manager panels, do either of the following:

- On the Primary Panel, type an I on the command line, specify a schedule name in the NEWNAME field, and then press Enter. Press Enter again to confirm that you want to insert a new schedule; the new schedule name is moved up to the NAME field.
- From the Schedule List panel (see “Select a Schedule”), type an I in the SEL column, and then type the name of the new schedule in the New Schedule field of the same line. Press Enter. Press Enter again to confirm that you want to insert a new schedule. The new schedule name is added to the schedule list.

  Schedule names have the format:

  ```
  sysname>schedname
  ```

  sysname

  Specifies the name of the system on which the schedule resides

  schedname

  Specifies the unique name for the schedule. If you do not specify a system name, > character, or both the current system name and > are automatically added as a prefix to the schedule name.

When you first create a schedule, system resources are associated with it but the schedule contains only the default period until you change it or define new periods.

Edit a Schedule

The two ways to edit a schedule are:

From the Primary Panel

1. Type the name of the schedule you want to edit in the NAME field.
2. Type E on the command line and press Enter.

From the Schedule List Panel

1. Type an E in the SEL column next to the name of the schedule you want to edit.
2. Press Enter to edit the schedule.

The Links Control panel is displayed.

Understand the Links Control Panel

The Links Control panel of the Schedule Manager controls basic editing functions for schedules.

You can use the Links Control panel, shown next, to do the following:

- Define or delete the periods controlling when System State Manager alters the desired states of resources.
- Specify which resources have their desired states changed at a given time.
- Select the new desired states that these resources will have.

You can return to the Links Control panel of the schedule you are editing at any time by issuing the following primary command:

```
[SHOW] LINK[S]
```

The following is a sample Links Control panel:

```
Schedule Manager ---------------- Links Control ----------------- System:OPS44R
Command ===>                                                    Scroll ===> CSR

CMDS: A D I L LC LD LX + ++ - --              |
-------------------------------------------- ------------------------------
    -PRODUCTION                              |     -ACF2_STCTBL
     |-WENDLITE    S.....S          0800-2000 |      |-STASK
     |-WENDDARK    S.....S          2000-3200 |      | |-ACF2
     |-EVERYNOON   SMTWTFS          1200-1300 |     -CICS_STCTBL
     |-WEEKDAY     .MTWTF.          0800-1600 |      |-STASK
     |-WEEKEVENING .MTWTF.          1600-2400 |      | |-CICS
     |-WEEKNIGHT   .MTWTF.          2400-3200 |      | |-CICSTEST
     |-THNXLITE03  11/27/2007-11/28 0800-2000 |     -DB2_STCTBL
     |-THNXDARK03  11/27/2007-11/28 2000-3200 |      |-STASK
     |-XMASLITE03  12/24/2007-12/25 0800-2000 |      | |-DB2
     |-XMASDARK03  12/24/2007-12/25 2000-3200 |     -IMS_STCTBL
     |-XMASALL03   12/25/2007       0000-2400 |      |-STASK
     |-XMASEVERY   12/25            0000-2400 |      | |-IMS
     |-DEFAULT                                |     -JES2_STCTBL
                                              |      |-STASK
                                              |      | |-JES2
                                              |     -TSO_STCTBL
                                              |      |-STASK
                                              |      | |-TCAS
                                              |      | |-TCASTEST
                                              |
(4)    (1)                (2)                |        (3)
```

Each of the following numbered paragraphs describes the fields on the Links Control panel designated with a number in bold:

1. The Period Names Column
2. The period names column contains the name of the schedule being edited (-PRODUCTION in this example) followed by the names of periods belonging to the schedule.
3. The Period Definitions Column

   The period definitions column displays when the period is active (that is, the days of the week or dates that the period is active, followed by the period start and stop times).

   A seven-character combination of dots and letters is used to display the days of the week when a cyclic period is active. The first character represents Sunday, the second character represents Monday, and so on, with the last character representing Saturday. If a period is to be active on a given day, then you see the first letter of the name of the day. A dot indicates that the period does not take effect on that day. Using this example, period WENDLITE is active on Sunday and Saturday (S.....S), period WEEKDAY is active Monday, Tuesday, Wednesday, Thursday, and Friday (.MTWTF.), and period EVERYNOON is active every day of the week (SMTWTFS).

   Notes:

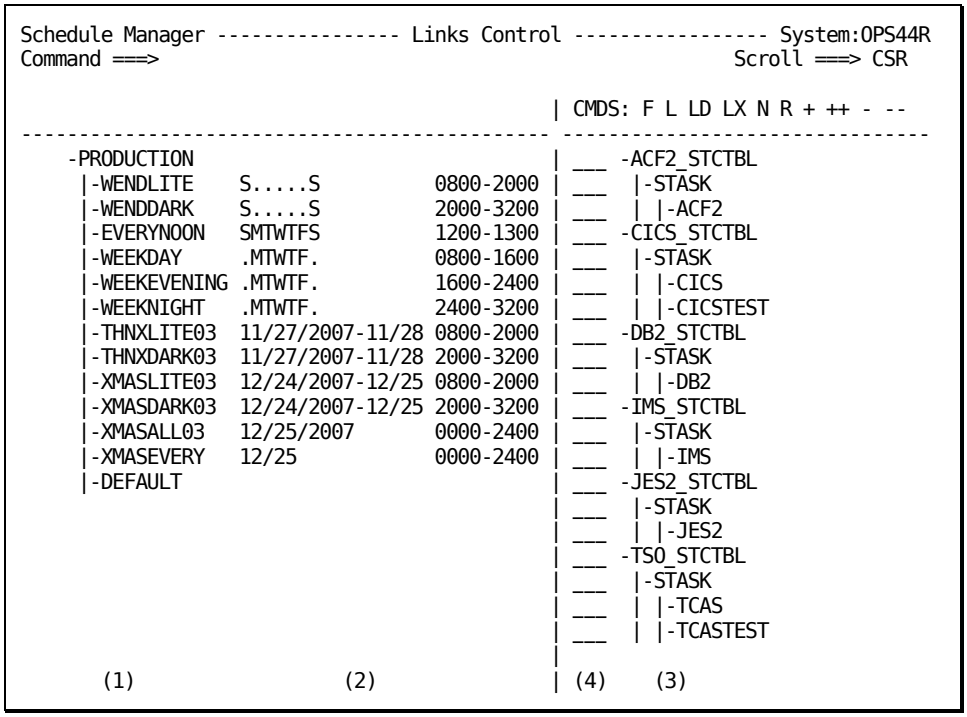
   - Fixed date periods may (such as period XMASLITE in this example) or may not (such as period NEWYEARLITE in this example) have a year specified (see Defining a Fixed Dates Period in this chapter.)
   - Periods can go beyond midnight into the next day. Using this example, period WENDDARK is active from 10:00 p.m. (2000) on Saturday until 8:00 a.m. (3200) on Sunday, and from 10:00 p.m. (2000) on Sunday until 8:00 a.m. (3200) on Monday. For more information, see The Time Interval for a Period in this chapter.
4. The Resources Column

   The resources column contains nested information about your system resources. The Schedule Manager gets this information from the SSM_MANAGED_TBLS table. The values starting in the leftmost position are the names of the System State Manager tables, such as table -ACF2_STCTBL in this example. The values nested one level below the table names are the names of resource types, such as type -STASK in this example. The values nested at the lowest level are resource names, such as resource -ACF2 in this example.

5. The CMDS Column

   The CMDS column contains fields for entering the line commands listed on the CMDS line of the panel. Line commands can be issued against either periods or resources by positioning the CMDS column in the appropriate place on the panel. To position the CMDS column to enter the commands against periods, press the PF10 key or issue the LEFT command. This is the default display, shown in the previous example.

   To position the CMDS column to issue line commands against resources, press the PF11 key or issue the RIGHT command. The following is an example of the Links Control panel after the PF11 key was pressed or the RIGHT command was issued to toggle the CMDS column. Note that the list of valid line commands changes according to where the CMDS column is positioned.



```
Schedule Manager ---------------- Links Control ----------------- System:OPS44R
Command ===>                                                  Scroll ===> CSR

                                            | CMDS: F L LD LX N R + ++ - --
--------------------------------------------- -------------------------------
    -PRODUCTION                             | ___ -ACF2_STCTBL
     |-WENDLITE    S.....S          0800-2000 | ___  |-STASK
     |-WENDDARK    S.....S          2000-3200 | ___  | |-ACF2
     |-EVERYNOON   SMTWTFS          1200-1300 | ___ -CICS_STCTBL
     |-WEEKDAY     .MTWTF.          0800-1600 | ___  |-STASK
     |-WEEKEVENING .MTWTF.          1600-2400 | ___  | |-CICS
     |-WEEKNIGHT   .MTWTF.          2400-3200 | ___  | |-CICSTEST
     |-THNXLITE03  11/27/2007-11/28 0800-2000 | ___ -DB2_STCTBL
     |-THNXDARK03  11/27/2007-11/28 2000-3200 | ___  |-STASK
     |-XMASLITE03  12/24/2007-12/25 0800-2000 | ___  | |-DB2
     |-XMASDARK03  12/24/2007-12/25 2000-3200 | ___ -IMS_STCTBL
     |-XMASALL03   12/25/2007       0000-2400 | ___  |-STASK
     |-XMASEVERY   12/25            0000-2400 | ___  | |-IMS
     |-DEFAULT                                | ___ -JES2_STCTBL
                                              | ___  |-STASK
                                              | ___  | |-JES2
                                              | ___ -TSO_STCTBL
                                              | ___  |-STASK
                                              | ___  | |-TCAS
                                              | ___  | |-TCASTEST
                                              |
       (1)                 (2)                | (4)    (3)
```

The Meaning of Display Colors

Field display colors used on the Schedule Manager panels help you to distinguish which periods are linked to which resources.

Blue

Periods and resources usually appear on the screen in blue.

White

When you issue the L line command (see Line Commands on the Links Control Panel in this chapter) against a period or a resource, that period or resource becomes the *link item* and its display color changes to white.

Green

The resources linked to the selected period (or the periods linked to the selected resource) become the *link group* and their display color changes to green or red. Green indicates a link group item that is supposed to be active (or online, up, and so on) for this period or resource pair.

Red

Red indicates a link group item that is supposed to be inactive (or offline, down, and so on).

Control What Appears Onscreen

You can expand and collapse the period or resource data trees displayed on the Links Control panel by issuing the following line commands:

Line command	Description
+	Expands the information tree one level
++	Expands the information tree two levels
-	Collapses the information tree one level
--	Collapses the information tree two levels

These commands are most useful on the resource tree.

The following example shows the above panel after line command -- was entered to collapse table -TSO_STCTBL and line command - was entered to collapse the -STASK types in table -CICS_STCTBL:

```
Schedule Manager ---------------- Links Control -------------- System:OPS11.7R
Command ===>                                                    Scroll ===> CSR

                                                | CMDS: F L LD LX N R + ++ - --
------------------------------------------------ -------------------------------
    -PRODUCTION                                 | ___ -ACF2_STCTBL
     |-WENDLITE    S.....S          0800-2000 | ___  |-STASK
     |-WENDDARK    S.....S          2000-3200 | ___  | |-ACF2
     |-EVERYNOON   SMTWTFS          1200-1300 | ___ -CICS_STCTBL
     |-WEEKDAY     .MTWTF.          0800-1600 | ___  |+STASK
     |-WEEKEVENING .MTWTF.          1600-2400 | ___ -DB2_STCTBL
     |-WEEKNIGHT   .MTWTF.          2400-3200 | ___  |-STASK
     |-THNXLITE03  11/27/2007-11/28 0800-2000 | ___  | |-DB2
     |-THNXDARK03  11/27/2007-11/28 2000-3200 | ___ -IMS_STCTBL
     |-XMASLITE03  12/24/2007-12/25 0800-2000 | ___  |-STASK
     |-XMASDARK03  12/24/2007-12/25 2000-3200 | ___  | |-IMS
     |-XMASALL03   12/25/2007       0000-2400 | ___ -JES2_STCTBL
     |-XMASEVERY   12/25            0000-2400 | ___  |-STASK
     |-DEFAULT                                | ___  | |-JES2
                                              | ___ +TSO_STCTBL
                                              |
       (1)                 (2)                | (4)    (3)
```

The following is an example of the Links Control panel after the line command ++ was issued to expand table -TSO_STCTBL and line command + was issued to expand the STASK types in table -CICS_STCTBL:

```
Schedule Manager ---------------- Links Control ----------------- System:OPS11.7R
Command ===>                                                     Scroll ===> CSR

                                                | CMDS: F L LD LX N R + ++ - --
------------------------------------------------ -------------------------------
    -PRODUCTION                                 | ___ -ACF2_STCTBL
     |-WENDLITE    S.....S          0800-2000 | ___  |-STASK
     |-WENDDARK    S.....S          2000-3200 | ___  | |-ACF2
     |-EVERYNOON   SMTWTFS          1200-1300 | ___ -CICS_STCTBL
     |-WEEKDAY     .MTWTF.          0800-1600 | ___  |-STASK
     |-WEEKEVENING .MTWTF.          1600-2400 | ___  | |-CICS
     |-WEEKNIGHT   .MTWTF.          2400-3200 | ___  | |-CICSTEST
     |-THNXLITE03  11/27/2007-11/28 0800-2000 | ___ -DB2_STCTBL
     |-THNXDARK03  11/27/2007-11/28 2000-3200 | ___  |-STASK
     |-XMASLITE03  12/24/2007-12/25 0800-2000 | ___  | |-DB2
     |-XMASDARK03  12/24/2007-12/25 2000-3200 | ___ -IMS_STCTBL
     |-XMASALL03   12/25/2007       0000-2400 | ___  |-STASK
     |-XMASEVERY   12/25            0000-2400 | ___  | |-IMS
     |-DEFAULT                                | ___ -JES2_STCTBL
                                              | ___  |-STASK
                                              | ___  | |-JES2
                                              | ___ -TSO_STCTBL
                                              | ___  |-STASK
                                              | ___  | |-TCAS
                                              | ___  | |-TCASTEST
                                              |
       (1)                 (2)                | (4)    (3)
```

Line Commands on the Links Control Panel

The CMDS line of the Links Control panel displays a set of line commands that let you create and modify periods, link resources with periods, and determine which periods and resources the Schedule Manager displays on your screen.

The following provides a description of each line command:

A

Lets you alter a period definition, changing the name, times and days, or both associated with an existing period. You can also use the A line command to replace the group of resources currently linked to a period with the link group from another existing period.

For information about defining periods, see Defining a Period in this chapter.

D

Deletes a period definition.

F or OFF

Operating only on items in the link group column, the F line command or the OFF line command creates a link between a resource and a period that turns the resource OFF during that period. The display color of the item against which you issue the command changes to red.

If an existing link between this resource and period specifies the desired state of the resource as ON or its equivalent, then the F line command or OFF line command changes the desired state to OFF or its equivalent. If you issue either command against the name of a table or a resource type, then the Schedule Manager creates links that set to OFF (or its equivalent) the desired states of the resources of that table or that type.

I

Lets you define a new period. For information about defining periods, see Defining a Period in this chapter.

L

Issuing this command against a period or resource makes that period or resource the current Link Item, and makes the items that are linked to that period or resource the Link Group. If the link item is a period, then the Schedule Manager displays all resources linked to that period, and each resource appears in red or green depending on whether the link specifies that the resource is ON or OFF during the period. If the link item is a resource, then you see all periods linked to that resource and the periods appear in red or green as appropriate.

LC

Issuing the LC line command against a resource or period copies the links of the current Link Item to the item against which the LC was issued. The LC line command is only available from the Link Item Column.

LD

Issuing the LD line command against a resource or period deletes all the links associated with that item.

LX

Issuing the LX line command against an item makes that item the Link Item and displays only the resources or periods linked to that item.

Note: Hidden items can be redisplayed using the +/++ or L line commands.

If the link item is a period, then the Schedule Manager displays:

- The names of all active System State Manager tables, shown in blue.
- The names of types that have one or more resources linked to the chosen period, shown in blue.
- The resources associated with the period, in red or green.

If the link item is a resource, then the Schedule Manager displays only:

- The name of the schedule, shown in blue.
- The names of the periods linked to the resource, in red or green.

N or ON

Operating only on items in the link group column, the N line command or the ON line command creates a link between a resource and a period that turns the resource ON during that period. The display color of the item against which you issue the command changes to green.

If an existing link between this resource and period specifies the desired state of the resource as OFF or its equivalent, then the N line command or the ON line command changes the desired state to ON or its equivalent. If you issue either command against the name of a table or a resource type, then the Schedule Manager creates links that set to ON (or its equivalent) the desired states of all of the resources of that table or that type.

R

Issuing the R line command against a period or a resource removes a link between that item and the current link item. If you issue the R line command against a table or a resource type, then the Schedule Manager removes all links for the resources of that table or type and changes their display color to blue. You can issue the R line command only in the link group column.

+, -, ++, and --

For descriptions of these line commands and what they do, see Controlling What Appears Onscreen in this chapter.

Define a Period

To define a period

1. Issue the I line command from the Links Control panel.

   The Schedule Manager displays its Period Maintenance Panel, shown here:

```
Schedule Manager ----------- Period Maintenance Panel ----------- System:OPS11.7R
Command ===>

Period Name: __________  (Required. Up to eleven alphanumeric characters)
Start Time : ____        (Required. HHMM - Any value from 0000 to 4800)
End   Time : ____        (Required. HHMM - Any value from Start Time to 4800)

Specify values for ONLY ONE of the following types of period definitions:

1. Place an S beside each day of the week that this period will be active:
     _ Sunday _ Monday _ Tuesday _ Wednesday _ Thursday _ Friday _ Saturday

2. Specify a first (and optional last) day of the year that this period will be
   active. (Period will only be active First Date if Last Date not specified.)
     First Date ===> Month: __    (1 - 12)
                     Day  : __    (1 - last day of month)
     Optional        Year : ____  (Period active EVERY YEAR if not specified)
      Last Date ===> Month: __    (1 - 12)
                     Day  : __    (1 - last day of month)

_ Select this option to make the set of links for this period a DUPLICATE of
  the set of links for the period named: __________
```

   - The Default Period of the Schedule Manager

     The set of periods that applies to a certain time may not describe the desired state of all resources on your system, and you may not have defined periods to cover all times. So, the Schedule Manager uses a period called DEFAULT to define the desired states of those resources whose desired states are not defined by any other period.

     If the Schedule Manager cannot find a resource in the set of periods that control a certain point in time or in the DEFAULT_STATE period, then the Schedule Manager does not set the desired state of the resource.

   - The Period Names

     The period name can contain 1 to 11 characters and must start with an alphabetic or national character. National characters for the United States are $, @, and #.

2. Specify the time interval that this period covers as hours and minutes.

   In most cases, specify this time in military time on a 24-hour clock, without a colon between hours and minutes (for example, 10:00 p.m. becomes 2200). However, if the time interval for a period starts before midnight and ends after midnight, you must specify the time on a 48-hour clock with 2400 representing midnight. For instance, a start time of 2300 and an end time of 3700 specifies a time interval lasting from 11:00 a.m. until 1:00 p.m. the following afternoon.

   If a resource must be UP or DOWN continuously over two consecutive periods, then make the end time of the earlier period and the start time of the later period exactly the same. Otherwise, System State Manager may bring the resource UP or DOWN for the interval between when the earlier period expires and the next period becomes active.

3. Define a cyclic days of the week period.

   To specify each day of the week that the period should be active, type an S beside the name of that day on the Period Maintenance Panel. For example, to make a period active Monday through Wednesday and on Saturday, type an S beside Monday, Tuesday, Wednesday, and Saturday.

   The cyclic days of the week period are defined.

4. Define a fixed dates period.

   - To specify a single date when the period should be active, enter numeric values for month, day, and optional year for the First Date field. Numeric values are automatically padded to the left as necessary. For example, you can specify the month of February as 2; the 4th day of the month as 4; and the year 2009 as 9.
   - To specify a range of dates when the period should be active, enter month and day values for both the First Date and Last Date fields. Enter an optional year value for the First Date field only; the year for the Last Date field is assumed from the First Date field. A single period cannot span more than one year. The first date can be later in the year than the last date. If it is, then the period remains active into the following year. For example, a period with a first date of April 12 and a last date of April 10 is active every day of the year except April 11.

     If the optional year is not specified, then the period is active every year on the specified dates (unless the start date is February 29, in which case the period is active only in leap years).

Resolving Conflicts

When two periods overlap in time with one period containing an active link and the other period containing an inactive link to the same resource, then those periods are said to be in conflict. For a description of how the Schedule Manager resolves schedule conflicts, see How Schedule Manager Resolves Schedule Conflicts in this chapter.

When a fixed period and a cyclic period are in conflict over scheduled control of a common resource, the Schedule Manager gives control to the fixed period.

When a fixed period with a year specified and a fixed period without a year specified are in conflict over scheduled control of a common resource, the Schedule Manager gives control to the fixed period with the year specified.

The SHOW CONFLICTS command (see View Schedule Conflicts in this chapter), the SHOW OVERLAPS command (see View Schedule Overlaps in this chapter), and the REPORT CONFLICTS command (see The REPORT Primary Command in this chapter) all report only on conflicts between periods not resolved by the above rules (that is, between two cyclic periods or between two fixed periods both with or without the year specified). Those conflicts are resolved by start time (see How Schedule Manager Resolves Schedule Conflicts in this chapter).

Any schedule can contain periods that span the same point in time. In other words, more than one period may be active at any given time. Periods that are active at the same time are said to overlap. A schedule overlap may be created intentionally or unintentionally. In either case, the affect of the overlap may be good or bad. For a discussion on the affect of overlaps, see Viewing Schedule Conflicts in this chapter.

Note: Schedule Manager brings resources defined for a period up when that period begins, but does not take those resources down when that period ends, in case a subsequent or overlapping period needs those resources to be active. In addition:

- The definition of the default period for your site must include links to all resources under the control of the Schedule Manager.
- The periods you define should keep control of the desired state of each resource continuously throughout all 24 hours of a day. You can define multiple periods to accomplish this, but at minimum you should define the desired state of a resource for the default period.

Period Overlap Conflicts

When two periods overlap in time with one period containing an active link and the other period containing an inactive link to the same resource, then those periods are said to be in conflict.

- When a fixed period and a cyclic period are in conflict over scheduled control of a common resource, the Schedule Manager gives control to the fixed period.
- When a fixed period with a year specified and a fixed period without a year specified are in conflict over scheduled control of a common resource, the Schedule Manager gives control to the fixed period with the year specified.

The SHOW CONFLICTS command (see View Schedule Conflicts in this chapter), the SHOW OVERLAPS command (see View Schedule Overlaps in this chapter), and the REPORT CONFLICTS command (see The REPORT Primary Command in this chapter) all report only on conflicts between periods not resolved by the above rules (that is, between two cyclic periods or between two fixed periods both with or without the year specified). Those conflicts are resolved by start time.

Any schedule can contain periods that span the same point in time. In other words, more than one period may be active at any given time. Periods that are active at the same time are said to overlap. A schedule overlap may be created intentionally or unintentionally. In either case, the affect of the overlap may be good or bad.

Note: Schedule Manager brings resources defined for a period up when that period begins, but does not take those resources down when that period ends, in case a subsequent or overlapping period needs those resources to be active. In addition:

- The definition of the default period for your site must include links to all resources under the control of the Schedule Manager.
- The periods you define should keep control of the desired state of each resource continuously throughout all 24 hours of a day. You can define multiple periods to accomplish this, but at minimum you should define the desired state of a resource for the default period.

More information:

View Schedule Conflicts (see page 352)
How Schedule Manager Resolves Schedule Conflicts (see page 354)
View Schedule Overlaps (see page 351)
REPORT Command—Print Schedule Manager Data (see page 355)

Change a Period Definition

To edit a period definition

1. Place the cursor on the line for that period, type A in the CMDS column, and press Enter.

   The Period Maintenance Panel appears.

2. Make the changes you want by typing over the values in the appropriate input fields.

3. Press Enter.

Your changes are saved and you return to the Links Control panel.

Delete a Period

To delete a period from a schedule

1. From the Links Control panel, place the cursor on the line for that period.

2. Type D in the CMDS column.

3. Press Enter.

The deleted period disappears from the screen.

Establish Unique Links

Links between periods and system resources determine the following:

- Which resources have their desired state controlled when a period is active
- Whether the desired state is set to ON (or equivalent statuses such as UP, ACTIVE, ONLINE, and so on), to OFF (or equivalent statuses such as DOWN, INACTIVE, OFFLINE, and so on), or is not set at all

To establish a unique set of links between a period and various system resources

1. On the Links Control panel, issue the L line command in the command column on the line describing a period.

   The selected period becomes the link item and its display color changes to white.

2. Press the PF11 key to move the CMDS column into the resource column.

3. Look at the list of resources, resource types, and tables displayed in the resources column.

   Note: Press the PF7 key to scroll up or the PF8 key to scroll down.

   Decide which resources you want to link this period to and what you want the desired states of those resources to be. A period can be linked to as many resources as you want, including resources belonging to different resource types or different tables.

4. Place the cursor on the line beside the item you want to link to the currently selected period.
   - To link to a single resource, place the cursor on the line listing that resource.
   - To link to all resources under the same resource type, place the cursor on the line listing that type.
   - To link to all resources belonging to the same table, place the cursor on the line listing the table name.
5. Type either F (off) or N (on) in the CMDS column.
   - Type F if you want the resource or resources in this link to have a desired state of OFF or its equivalent during the current period.
   - Type N if you want the resource or resources to have a desired state of ON or its equivalent during the current period.
6. Press Enter.

   The Schedule Manager links the selected resource or resources to the current period and saves the link, making it part of the current schedule. The resources you chose turn green if their desired state is ON or its equivalent, or red if their desired state is OFF or its equivalent.
7. Repeat the previous two steps if you want to create links to other resources.

Define the Same Links for Several Periods

This step gives a period the same links to resources as another period.

To define the same links

1. On the Links Control panel, type an A next to the period name and press Enter.

   The Period Maintenance Panel displays.
2. On the Period Maintenance Panel, type an S beside the paragraph beginning with Select this option.
3. Type the name of another period.

   The period you are defining will have the same links to resources as the period whose name you enter in this step.
4. Press Enter.

   The Schedule Manager deletes all existing links for this period, duplicates the links of the period whose name you specified, and returns you to the Links Control panel.

Delete a Period/Resource Link

To delete a link between a period and one or more resources:

1. Place the cursor on the line for the period, type L in the CMDS column, and press Enter.

   The selected period becomes the link item, and its display color becomes white.

2. Press the PF11 key to move the CMDS column to the resource column.

3. Place the cursor beside the resource, the resource type, or the table to which you want to dissolve the link.

4. Type R and press Enter.

   The Schedule Manager removes the link or links to the selected resource or the resources in the selected type or table. The display color for these resources changes from red or green to blue.

Note: You can also delete a period/resource link by typing L next to the resource, pressing the PF10 key, typing R next to the period, and pressing Enter.

Activate (Load) a Schedule

System State Manager always draws its scheduling information from the schedule named ACTIVE.

The ACTIVE schedule sets the desired states of resources according to the links set for each period.

The two ways to load a schedule are:

- From the Primary Panel
- From the Schedule List Panel

To load a schedule from the Primary Panel

1. Type the name of the schedule to activate in the NAME field.
2. Type L on the command line and press Enter.
3. Press Enter again to load the schedule.

To load a schedule from the Schedule List Panel

1. Type an L in the SEL column next to the name of the schedule you want to activate.
2. Press Enter, and then Enter again to confirm that you want to load the selected schedule.

Result-The schedule you selected is copied into the ACTIVE schedule, allowing System State Manager to use its data to determine when and how the desired states of resources are to be set.

SHOW STATES Command—View the Scheduled States of Resources

The Schedule Manager allows you to display the scheduled state of all your system resources at a given time and date. To do so, issue the SHOW STATE primary command from the command line.

This command has the following format:

```
[SHOW] STAT[ES] [AT hhmm ON [mm/dd/yyyy]]]
```

hhmm

Specifies the time values. Specify this time value in military time (for example, use 1500 for 3:00 p.m.).

mm/dd/yyyy

Specifies the date values.

If you specify no *hhmm* time or *mm/dd/yyyy* date values, then the Schedule Manager shows you the scheduled system state at the current date and time. If you specify only a time, then the Schedule Manager shows you the scheduled system state at that time today. If you specify a *mm/dd* date value without the optional */yyyy* year value, then the Schedule Manager shows you the system state at the specified *hhmm* time on the next occurrence of the specified *mm/dd* date.

The following examples assume that the current time is 3:00 p.m. on February 21, 2010:

- This command will show the system state as it is now at 3:00 p.m. today, on February 21, 2004:

  ```
  SHOW STATE
  ```

- This command will show the system state as it was at 1:00 p.m. today, on February 21, 2004 (assuming that the schedule has not changed since then):

  ```
  SHOW STATE AT 1300
  ```

- This command will show the system state as it will be at 1:00 p.m. tomorrow, on February 22, 2004:

  ```
  SHOW STATE AT 1300 ON 02/22
  ```

- This command will show the system state as it was at 2:00 p.m. yesterday, on February 21, 2004 (assuming that the schedule has not changed since then):

  ```
  SHOW STATE AT 1400 ON 2/21/2004
  ```

- This command will show the system state as it will be at 2:00 p.m. a year from yesterday, on February 21, 2005:

  ```
  SHOW STATE AT 1400 ON 2/21
  ```

Given the sample schedule PRODUCTION used in Understanding the Links Control Panel in this chapter, the following panel would appear if the command SHOW STATE AT 1400 ON 2/21 was issued:

```
Schedule Manager ----- State at 1400 on 02/21/2005 (MON) -------- System:OPS44R

Command ===>                                                  Scroll ===> CSR

                                          | CMDS: F N R + ++ - --
--------------------------------------------- --------------------------------
                                          - ___ -ACF2_STCTBL
                                          - ___  |-STASK
    WEEKDAY        .MTWTF.          0800-1600 - ___  | |-ACF2
                                          - ___ -CICS_STCTBL
                                          - ___  |-STASK
    WEEKDAY        .MTWTF.          0800-1600 - ___  | |-CICS
    WEEKDAY        .MTWTF.          0800-1600 - ___  | |-CICSTEST
                                          - ___ -DB2_STCTBL
                                          - ___  |-STASK
                                          - ___  | |-DB2
                                          - ___ -IMS_STCTBL
                                          - ___  |-STASK
    WEEKDAY        .MTWTF.          0800-1600 - ___  | |-IMS
                                          - ___ -JES2_STCTBL
                                          - ___  |-STASK
    WEEKDAY        .MTWTF.          0800-1600 - ___  | |-JES2
                                          - ___ -TSO_STCTBL
                                          - ___  |-STASK
    WEEKDAY        .MTWTF.          0800-1600 - ___  | |-TCAS
    WEEKDAY        .MTWTF.          0800-1600 - ___  | |-TCASTEST
```

The above example shows the State panel that displays the system state information. The title of the panel and the separator between the period and resource columns change to remind you that you are seeing scheduled states of resources instead of period/resource links. The periods no longer are displayed in a tree structure; instead, in each row you see the period that is controlling the resource in that row at the time shown on the title line.

Note: The period column is blank if the resource is not controlled by any period at the given date and time.

When displaying the State panel, you can issue commands only against resources, and you can issue only the commands shown. The +, ++, -, and -- line commands work as they do when you display links. The F and N line commands work as they do when you display links except that:

- The commands change only the state of the resource in the period on that row.
- When you issue the N line command or the F line command against a resource not controlled by any period, you see a list of possible periods to link the resource to and are asked to select a period.

The R line command, when issued against a resource associated with a period, removes that resource from the period. If no other period is linked to that resource, then the period column becomes blank and the resource display color changes to blue. But, if there is a link for that resource in another period that covers the time displayed on the title line:

- The period name in the row where you issued the R line command becomes the name of the other period, and
- The display color for the resource changes to match the link definition for the other period

The links in the previous sample display would be color-coded-green for active links and red for inactive links. In a black and white environment such as this guide, color-coding is not sufficient to identify the direction of the displayed links.

More information:

The Meaning of Display Colors (see page 326)

CMDSONOFF Primary Command—Distinguish Active and Inactive Links

In addition to the field display colors green and red, you can have the Schedule Manager include the words ON and OFF on the CMDS column to help you distinguish between active and inactive links.

The primary command CMDSONOFF accepts the following keyword settings to control this feature:

ON

Green (for active) links contain the word ON in the CMDS column and red (for inactive) links contain the word OFF in the CMDS column.

OFF

The keywords ON and OFF are not used for display purposes in the CMDS field.

Default: OFF

After issuing the command CMDSONOFF ON, the previous panel would look similar to the one below:

```
Schedule Manager ----- State at 1400 on 02/21/2005 (MON) -------- System:OPS44R
ommand ===>                                                    Scroll ===> CSR

                                          | CMDS: F N R + ++ - --
--------------------------------------------- -------------------------------
                                          - ___ -ACF2_STCTBL
                                          - ___  |-STASK
    WEEKDAY        .MTWTF.          0800-1600 - ON_  | |-ACF2
                                          - ___ -CICS_STCTBL
                                          - ___  |-STASK
    WEEKDAY        .MTWTF.          0800-1600 - ON_  | |-CICS
    WEEKDAY        .MTWTF.          0800-1600 - OFF  | |-CICSTEST
                                          - ___ -DB2_STCTBL
                                          - ___  |-STASK
                                          - ___  | |-DB2
                                          - ___ -IMS_STCTBL
                                          - ___  |-STASK
    WEEKDAY        .MTWTF.          0800-1600 - ON_  | |-IMS
                                          - ___ -JES2_STCTBL
                                          - ___  |-STASK
    WEEKDAY        .MTWTF.          0800-1600 - ON_  | |-JES2
                                          - ___ -TSO_STCTBL
                                          - ___  |-STASK
    WEEKDAY        .MTWTF.          0800-1600 - ON_  | |-TCAS
    WEEKDAY        .MTWTF.          0800-1600 - OFF  | |-TCASTEST
                                          -
```

View Potential State Changes

From the State panel, you can view the scheduled state of system resources at the next or previous scheduled period start or stop time. The next or previous scheduled period start or stop time is the next or previous time for a potential change to the scheduled state of system resources. Schedule Manager does not verify that there is an actual change in the scheduled state of any system resources at the time of potential change.

To view potential state changes

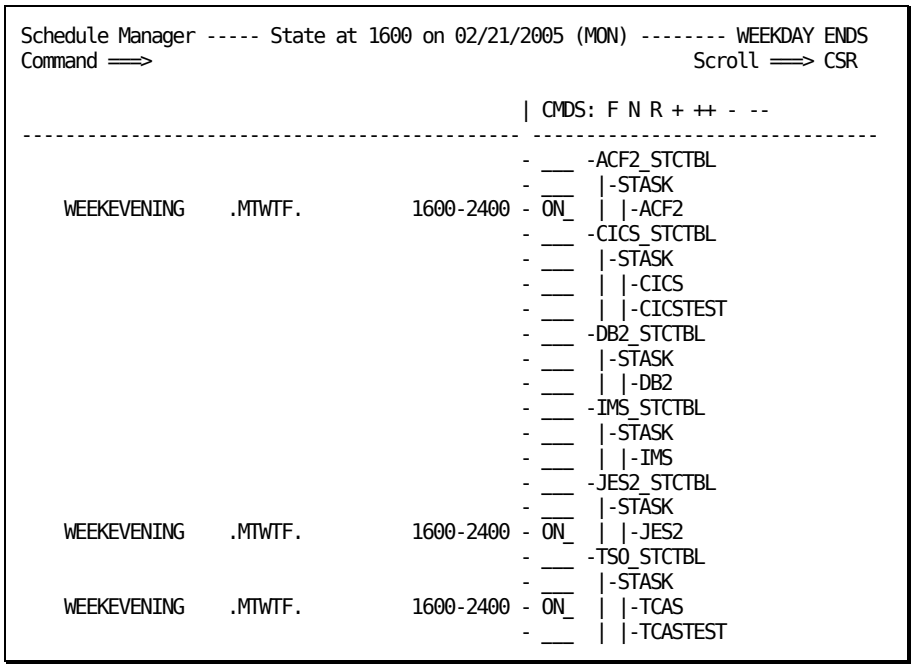
1. Issue the following command:

   ```
   [SHOW] NEXT
   ```

   The scheduled state of system resources at the time of the next scheduled period start or end displays.

2. From the State panel, press the PF11 key or issue the RIGHT command to automatically execute the command SHOW NEXT.

   After issuing a SHOW NEXT command from the State panel shown on the previous page, the following State panel would appear. Note the short help message in the upper right corner of the screen, indicating that the period has started or stopped (in this example, period WEEKDAY has stopped).



3. Issue the following commands to see you the scheduled state of system resources at the time of the most recent period start or stop that occurred:

   ```
   [SHOW] PREVious
   ```

   or

   ```
   [SHOW] PRIOR
   ```

4. Press the PF10 key or issue the command LEFT from the State panel to automatically execute the command SHOW PRIOR.

   After issuing a SHOW PRIOR command from the previous State panel, the State panel would look like the one below. Note that except for the time stamp, this display is identical to the first State panel sample display at time 1400. No periods start or stop between 1300 and 1400, therefore, no scheduled state changes can occur.

```
Schedule Manager ----- State at 1300 on 02/21/2005 (MON) -------EVERYNOON ENDS
Command ===>                                                  Scroll ===> CSR

                                          | CMDS: F N R + ++ - --
---------------------------------------------- -------------------------------
                                          - ___ -ACF2_STCTBL
                                          - ___  |-STASK
   WEEKDAY       .MTWTF.        0800-1600 - ON_  | |-ACF2
                                          - ___ -CICS_STCTBL
                                          - ___  |-STASK
   WEEKDAY       .MTWTF.        0800-1600 - ON_  | |-CICS
   WEEKDAY       .MTWTF.        0800-1600 - OFF  | |-CICSTEST
                                          - ___ -DB2_STCTBL
                                          - ___  |-STASK
                                          - ___  | |-DB2
                                          - ___ -IMS_STCTBL
                                          - ___  |-STASK
   WEEKDAY       .MTWTF.        0800-1600 - ON_  | |-IMS
                                          - ___ -JES2_STCTBL
                                          - ___  |-STASK
   WEEKDAY       .MTWTF.        0800-1600 - ON_  | |-JES2
                                          - ___ -TSO_STCTBL
                                          - ___  |-STASK
   WEEKDAY       .MTWTF.        0800-1600 - ON_  | |-TCAS
   WEEKDAY       .MTWTF.        0800-1600 - OFF  | |-TCASTEST
```

The Primary Commands in Edit

The following describes all of the primary commands available when editing a schedule:

CMDSONOFF OFF|ON

Controls the display of the words ON and OFF in the CMDS column in addition to the colors green and red when displaying active or inactive links. The default is OFF.

END

The default setting of the PF3 key. Enter this command to return to the Schedule List.

ENQTRACE OFF|ON

Controls tracing for use by CA technicians.

Default: OFF

LEFT

The default setting of the PF10 key. Issue this command from the Links Control Panel to toggle the CMDS column to the periods. Issue this command from the State panel to execute the SHOW PRIOR command.

NEXTRACE OFF|ON

Controls tracing for use by CA technicians.

Default: OFF

REPORT [*args*]

Print formatted schedule data to a data set. Optional *args* control what schedule data is printed and to where it is printed. See REPORT Command—Print Schedule Manager Data (see page 355).

RIGHT

The default setting of the PF11 key. Issue this command from the Links Control Panel to toggle the CMDS column to the resources. Issue this command from the State panel to execute the SHOW NEXT command.

[SHOW] CONF[LICTS]

Display in the Conflicts panel periods that overlap in time and contain at least one conflicting link to a common resource. For more information, see View Schedule Conflicts (see page 352).

[SHOW] LINK[S]

Display link items in the Links Control panel. Use the L line command to display link groups. This is the default display. For more information, see Understanding the Links Control Panel (see page 323).

[SHOW] NEXT

Display the next potential state change (period start or stop) after the state currently being displayed in the State panel. For more information, see View Potential State Changes (see page 342) in this chapter.

[SHOW] OVER[LAPS]

Display in the Overlaps panel periods that overlap in time. For more information, see View Schedule Overlaps (see page 351) in this chapter.

[SHOW] PREV[IOUS]

Issue the SHOW PRIOR command.

[SHOW] PRIOR

Display the last potential state change (period start or stop) before the state currently being displayed in the State panel. For more information, see View Potential State Changes (see page 342) in this chapter.

[SHOW] STAT[ES]

Display the scheduled state of resources in the State panel. For more information, see SHOW STATES Command—View the Scheduled State of Resources (see page 338) in this chapter.

STATRACE OFF|ON

Controls tracing for use by CA technicians.

Default: OFF

Copy a Schedule

The two ways to create a copy of a schedule are:

- From the Primary Panel
- From the Schedule List Panel

To create a copy of a schedule from the Primary Panel

1. Type the name of the schedule to be copied into the NAME field.
2. Type the name to assign to the copy in the NEWNAME field.
3. At the command line, type C and press Enter.
4. Press Enter again to confirm that you want to copy the schedule.

To create a copy of a schedule from the Schedule List Panel

1. Type a C in the SEL column next to the name of the schedule you want to copy.
2. Type the name of the target schedule in the New Schedule field on the same line.
3. Press Enter, and then Enter again to confirm that you want to copy that schedule.

Result-The Schedule Manager creates an exact duplicate of the named schedule and gives the copy the name specified in the NEWNAME/New Schedule field. The duplicate schedule contains the same periods and links to the same resources as the original schedule.

Rename a Schedule

The two ways to rename a schedule are from the following panels:

- Primary Panel
- Schedule List Panel

From the Primary Panel

1. Type the name of the schedule in the NAME field.
2. Type the new schedule name in the NEWNAME field.
3. Type R on the command line and press Enter.
4. Press Enter again to confirm that you want to rename the schedule.

From the Schedule List Panel

1. Type an R in the SEL column next to the name of the schedule you want to rename.
2. Type the new schedule name in the New Schedule field.
3. Press Enter, and then Enter again to confirm that you want to rename the schedule.

Result-The Schedule Manager replaces the schedule name with the name you specified.

Delete a Schedule

The two ways to delete a schedule are from the following panels:

- Primary Panel
- Schedule List Panel

From the Primary Panel

1. Type the name of the schedule to delete in the NAME field.
2. Type D on the command line and press Enter.

From the Schedule List Panel

1. Type a D in the SEL column next to the name of the schedule you want to delete.
2. Press Enter to delete the selected schedule.

Result-If the Confirm Deletes field on the Primary panel has the value YES, then you see a message prompting you to confirm your delete request. Press Enter again to confirm that you want to delete the specified file. Cancel a delete by typing END at the command line when asked to confirm a delete.

If you do not want to confirm delete requests, type NO over the YES value before deleting the schedule. The schedule is deleted without issuing a confirmation message.

Free a Schedule

System State Manager enqueues the processing of schedules. So, in case of failed links with cross-system MSF, or CAICCI cross-platform communications services or failed TSO address spaces, a user can own the enqueue for a schedule but be unable to free it for other users to access.

The two ways to free a schedule are from the following panels:

- Primary Panel
- Schedule List Panel

From the Primary Panel

1. In the NAME field, type the name of the schedule to be freed.
2. Type F on the command line and press Enter.

From the Schedule List Panel

1. Type an F in the SEL column next to the name of the schedule you want to free.
2. Press Enter to free the schedule.

Result-The F line command releases the schedule from the user who enqueued it, allowing other users access to that schedule.

Synchronize a Schedule

The two ways to synchronize a schedule are from the following panels:

- Primary Panel
- Schedule List Panel

From the Primary Panel

1. Type the name of the schedule you want to synchronize in the NAME field.
2. Type Y on the command line and press Enter.

From the Schedule List Panel

1. Type a Y in the SEL column next to the name of the schedule you want to synchronize.
2. Press Enter to synchronize the schedule.

Result-The Schedule Manager finds all links that point to periods that no longer exist or to resources that are no longer managed by System State Manager. Primarily, these invalid links are caused by removing a resource from System State Manager control without first removing it from Schedule Manager control; however, they can also be caused by system failures while a schedule is being edited. When an invalid link is found, the following menu of choices appears:

```
Schedule Manager ----------- SYNCH Selection List ---------- SELECT ONE ACTION
Command ===>                                                  Scroll ===> PAGE

Select one of the following actions for invalid TABLE:    SSMTEST

_ change all the links containing this item to use a new item name selected
  from the list of valid names shown below,

_ leave all the links containing this item as is,

_ delete all links containing this item,

_ stop showing this panel and leave as is all future links that contain an item
  that has not already had an action (change name/leave as is/delete) assigned,

_ stop showing this panel and delete all future links that contain an item that
  has not already had an action (change name/leave as is/delete) assigned.


Sel TABLE
___ SWZ_STCTBL
```

You must select one and only one action by typing an S next to the action and pressing the Enter key.

Merge Schedules

The two ways to merge a schedule are from the following panels:

- Primary Panel
- Schedule List Panel

From the Primary Panel

1. Type the name of the schedule to combine with another schedule in the NAME field.
2. Type the name of an existing schedule in the NEWNAME field.
3. Type M on the command line and press Enter.
4. Press Enter again to confirm that you want to merge the schedules.

From the Schedule List Panel

1. Type an M in the SEL column next to the name of the first schedule you want to merge.
2. In the New Schedule field, type the name of the second schedule you want to merge.
3. Press Enter, and then Enter again to confirm that you want to merge the schedules.

Result-The Schedule Manager merges the data from the two schedules into a schedule specified in the NEWNAME/New Schedule field.

If the names of the two schedules to be merged conflict, then Schedule Manager uses the name of the first schedule (the one listed in the NAME field). If the same period or link name appears in both schedules, then the data from the period specified in the NAME field overlays that of the period specified in the NEWNAME field.

View Schedule Overlaps

A schedule can contain periods that span the same point in time. In other words, more than one period can be active at any given time. Periods that are active at the same time are said to overlap. A schedule overlap may be created intentionally or unintentionally. In either case, the affect of the overlap may be good or bad. The affect of overlaps is discussed in the next section Viewing Schedule Conflicts.

Schedule Manager provides the following primary command to display scheduled period overlaps on the Overlaps panel:

```
[SHOW] OVERlaps
```

If no schedule overlaps exist, then you receive the short help message NO OVERLAPS in the upper right corner of the display you are currently in.

Issuing a SHOW OVERLAPS command while editing the sample schedule PRODUCTION would result in the following sample Overlaps panel being displayed:

```
Schedule Manager ------------------- Overlaps ------------------- System:OPS44R
Command ===>                                                  Scroll ===> CSR

CMDS: A D I L LC LD LX + ++ - --              |
------------------------------------------- ------------------------------
___ EVERYNOON      SMTWTFS          1200-1300 |     -ACF2_STCTBL
___ WENDLITE       S.....S          0800-2000 |      |-STASK
___                                           |      | |-ACF2
___ EVERYNOON      SMTWTFS          1200-1300 |     -CICS_STCTBL
___ WEEKDAY        .MTWTF.          0800-1600 |      |-STASK
___                                           |      | |-CICS
___ XMASLITE03     12/24/2007-12/25 0800-2000 |      | |-CICSTEST
___ XMASALL03      12/25/2007       0000-2400 |     -DB2_STCTBL
___                                           |      |-STASK
___ XMASDARK03     12/24/2007-12/25 2000-3200 |      | |-DB2
___ XMASALL03      12/25/2007       0000-2400 |     -IMS_STCTBL
                                              |      |-STASK
                                              |      | |-IMS
                                              |     -JES2_STCTBL
                                              |      |-STASK
                                              |      | |-JES2
                                              |     -TSO_STCTBL
                                              |      |-STASK
                                              |      | |-TCAS
                                              |      | |-TCASTEST
```

Similar to the Links Control panel, the Overlaps panel lists the SSM resource tree on the right side of the display. On the left side of the display it lists each schedule overlap separated by a blank line. For each overlap, the two overlapping period definitions are listed. Again, similar to the Links Control panel, line commands L and LX (listed on the CMDS line) are available to display defined links. Line commands C and CX are also available to display only links in conflict.

Note: The Overlaps panel does not display any overlaps between a cyclic day of week period and a fixed date period. For example, the above Overlaps panel displays overlaps between cyclic day of week period EVERYNOON and cyclic day of week periods WEEKLITE and WEEKDAY, but it does not display overlaps between cyclic day of week period EVERYNOON and any of the fixed date periods.

More information:

How Schedule Manager Resolves Schedule Conflicts (see page 354)
The C and CX Commands (see page 354)

View Schedule Conflicts

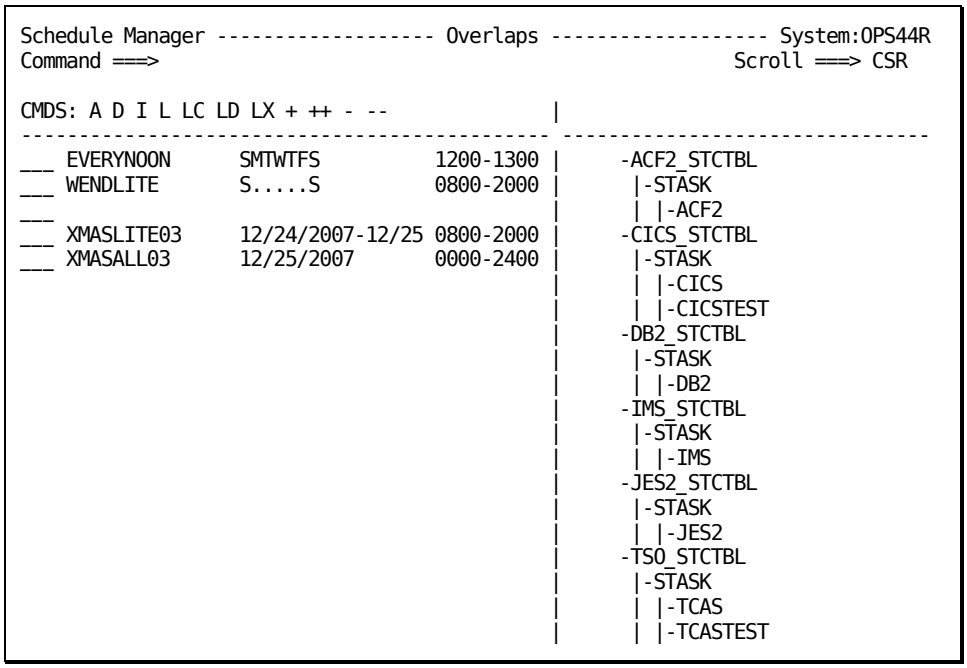
When two overlapping periods (discussed in the previous section Viewing Schedule Overlaps) each contain a link to the same resource, there may be a schedule conflict. A schedule conflict occurs when one of the links schedules the state of the resource to active and the other link schedules the state of the resource to inactive.

To identify schedule conflicts, issue the following command from the command line of the Links Control panel:

```
[SHOW] CONFlicts
```

If no schedule conflicts exist, then you receive the short help message NO CONFLICTS in the upper right corner of the display you are currently in.

Issuing a SHOW CONFLICTS command while editing the sample schedule PRODUCTION would result in the following sample Conflicts panel being displayed:



The Conflicts display of a schedule is identical to its Overlaps display, except that non-conflicting schedule overlaps are filtered out and only those schedule overlaps that contain conflicting links are listed.

The first period listed in each conflict pair is the period that sets the desired state of the resource.

Line commands L and LX are available to display defined links. Line commands C and CX are available to display only the links in conflict.

More information:

How Schedule Manager Resolves Schedule Conflicts (see page 354)

The C and CX Commands

From either the Overlaps or Conflicts panel, you can issue the following line commands against a link item (a period or resource) to display conflicting links:

C

Displays information about periods that specify conflicting resource states. This data includes information about all resources linked to the conflicting periods.

CX

Displays information about conflicting periods but excludes information about linked resources not involved in conflicts.

Schedule Manager excludes resource link items from the display by collapsing nodes on the resource tree (similar to the - and -- line commands). Excluded period link items are simply removed from the display.

How Schedule Manager Resolves Schedule Conflicts

When a cyclic day of week period and a fixed date period are in conflict over scheduled control of a common resource, Schedule Manager gives control to the fixed date period.

When a fixed date period *with* a year specified and a fixed date period *without* a year specified are in conflict over scheduled control of a common resource, Schedule Manager gives control to the fixed date period *with* the year specified.

Note: The SHOW CONFLICTS and SHOW OVERLAPS commands do not display any schedule conflicts resolved by the above rules. Also, the SHOW OVERLAPS command does not display any non-conflicting overlaps that would be resolved by the above rules if they were conflicts. This is done to make the displays more understandable and less complex.

Important! The user needs to keep this in mind when reading these displays.

Conflicts between periods not resolved by the above rules (that is, conflicts between two cyclic day of week periods or between two fixed date periods both with or without the year specified) are resolved as follows:

- The period with a start time closest to the time of conflict gains control of the resource. For instance, if period A takes effect Monday through Friday from 09:00 to 17:00 and period B takes effect Wednesday and Thursday from 10:00 to 22:00, then the times of conflict are between 10:00 and 17:00 on Wednesday and Thursday. Since 10:00, the start time of period B, is closer to the times of conflict than 09:00, the start time for period A, then period B controls the resource during the times of conflict between 10:00 and 17:00 on Wednesday and Thursday.
- If two conflicting periods have the same start time, then the period with a start date closest to the date of the conflict controls the resource. For example, if period A takes effect Monday through Friday from 09:00 to 12:00 and period B takes effect Wednesday and Thursday from 09:00 to 17:00, then period A controls the resource on Monday, Tuesday, and Friday between 09:00 and 12:00, and period B controls the resource from 09:00 until 12:00 on Wednesday and Thursday.
- If neither of the above criteria favors one period over the other, then the Schedule Manager uses the period whose name is first in alphabetical sequence. For instance, period A is used since A precedes B alphabetically.

REPORT Command—Print Schedule Manager Data

While editing a schedule, issue the REPORT commands at the command line to save to disk the same schedule information that is displayed by the SHOW commands.

Similar to the SHOW commands, the following are the REPORT commands:

REPORT

Report the SHOW display currently being viewed in Edit.

REPORT CLOSE

Close the report data set and stop it from receiving any more REPORT output.

REPORT CONFLICTS

Report conflicts between periods.

REPORT LINKS

Report links between periods and system resources.

REPORT OPEN [*dsn*]

Open the specified *dsn* or default report data set.

REPORT OVERLAPS

Report overlapping periods.

REPORT STATES . . .

Report the scheduled state of all your system resources.

Note: All of the command options available on the SHOW STATES command are also available on the REPORT STATES command.

Issuing the REPORT command by itself without any command options results in Schedule Manager reporting the SHOW display currently being viewed.

The first REPORT command to be issued with or without any command options creates a default report data set that is named using the following template:

```
hlq.SM.opssysid.schstrip.Dyyymmdd.Thhmm
```

Following is a description of the values in this template:

hlq

The default value returned by the standard OPDSPF() REXX routine, or the TSO userid of the user if OPSDSPF() returns null.

Note: If OPDSPF() is not customized by the user, then it returns a default value of either the TSO PREFIX setting, if one exists, or the TSO userid of the user.

opssysid

The system ID of the CA OPS/MVS system on which the schedule resides.

schstrip

The leftmost eight non-blank characters of the schedule name excluding any underscore characters.

yyymmdd

The current date without the high-order year digit (millennium).

hhmm

The current time of day in 24-hour military time.

You may optionally specify your own report data set name by issuing the following command first before issuing any other REPORT commands:

```
REPORT OPEN datasetname
```

The *datasetname* may be specified with or without quotes (' or "). If the *datasetname* is specified without quotes, then it is prefixed by *hlq* as described above.

Any errors that occur while attempting to allocate either the default or user-specified report data set result in the Dataset Creation Attributes panel being displayed. From there you are able to alter the data set name for allocation re-try by Schedule Manager, or SWAP into another ISPF screen to allocate the data set using ISPF, batch JCL, TSO commands, and so on.

After the first REPORT command completes, the output from any subsequent REPORT commands issued are appended to the end of the same report data set. The report data set is closed after each REPORT command completes and can be viewed, printed, copied, edited, and so on.

To close the report data set and stop it from receiving any more REPORT output, the user must issue the following command:

```
REPORT CLOSE
```

Note: Issue the HELP command after the REPORT CLOSE command returns to display the name of the report data set that was closed.

Example: REPORT Command

The next REPORT command to be issued after a REPORT CLOSE command creates a new report data set.

REPORT STATES AT 1300 ON 2/21/2005

The following would be the contents of data set JOEUSER.SM.OPS44R.PRODUCTI.D0030415.T1509. Note that this report is for the identical point in time that is shown in the example SHOW STATES display in Viewing Potential State Changes in this chapter.

```
Schedule: OPS44R>PRODUCTION ----------- STATES ----------- 15 Apr 2007
------------------------- AT 1300 ON 02/21/2005 (MON) -----------------

Table: ACF2_STCTBL
  Type: *NULL*
    ACF2               ON  WEEKDAY     .MTWTF.          0800-1600

Table: CICS_STCTBL
  Type: *NULL*
    CICS               ON  WEEKDAY     .MTWTF.          0800-1600
    CICSTEST           OFF WEEKDAY     .MTWTF.          0800-1600

Table: IMS_STCTBL
  Type: *NULL*
    IMS                ON  WEEKDAY     .MTWTF.          0800-1600

Table: JES2_STCTBL
  Type: *NULL*
    JES2               ON  WEEKDAY     .MTWTF.          0800-1600

Table: TSO_STCTBL
  Type: *NULL*
    TCAS               ON  WEEKDAY     .MTWTF.          0800-1600
    TCASTEST           OFF WEEKDAY     .MTWTF.          0800-1600
```

The REPORT LINKS command generates a report identical to that of a REPORT call to the Schedule Manager API. The next section describes the API in general and the REPORT call, and also includes a sample report.

Use the Schedule Manager Application Program Interface

In addition to using the Schedule Manager ISPF application through OPSVIEW primary option 4.11.4, a user can invoke Schedule Manager programmatically by calling the Schedule Manager Application Program Interface (API). This includes user-coded commands on the ISPF command line, user-coded programs running in batch JCL, user-coded programs running interactively under ISPF, user-coded AOF rules, and so on.

The CA OPS/MVS REXX program ASOSMAPI processes the Schedule Manager API calls. To use the Schedule Manager API, call ASOSMAPI in the proper environment using the proper command syntax as documented in this section.

API QUERY Command—Returns Schedule Manager Data

The API QUERY command returns Schedule Manager data, such as schedule names, period definitions, defined links, and so on. The data returned depends on the keywords entered on the QUERY command text. Syntax and examples of the numerous possible keyword combinations on the QUERY command are shown below.

Data is returned by the QUERY command in the following format:

```
delimit||code||delimit||count||delimit||value1||delimit||value2|
|delimit||...
```

In the format above, the first character returned is used to *delimit* values in the returned data. The first value returned is the return *code* of the QUERY operation. The second value returned is the *count* of data *values* that follow. The data always ends with the *delimit* character. When return *code* > 0, then *count* is actually the text of an error message. When return *code* = 0, then *count* may also be 0.

Examples given in this section were generated by calling the following REXX program named SM$APIQ to call the Schedule Manager API and display the results at the terminal:

```
/*--------------------------------------------------------------------*/
/*  REXX test ASOSMAPI API QUERY calls.                               */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+---*/
/*                                                                    */
  PARSE UPPER ARG sm_api_parms
  CALL ASOSMAPI sm_api_parms
  SAY 'SM$APIQ Result='result
IF result='RESULT' THEN DO
    say 'Bad QUERY'
  END
  ELSE DO
    PARSE VAR result rdelim 2 rcode (rdelim) rcount (rdelim) +0 rresult
    IF rcode<>0 THEN DO
      say 'QUERY returned error code "'||rcode||'" and error '
      say 'text: "'||rcount||'"'
    END
    ELSE DO i = 1 TO rcount
      PARSE VAR rresult rdelim 2 rentry (rdelim) +0 rresult
      SAY RIGHT(i,3)||':'||rentry
    END
  END
EXIT result
```

Program SM$APIQ above was called by issuing the following command from the ISPF command shell on a TSO user ID logged on to a system with a single copy of CA OPS/MVS running:

```
OI SM$APIQ arg
```

Argument *arg* in the above command text contains the QUERY command syntax to call Schedule Manager, documented below.

For more information on the OI command, see OPSIMEX Command Processor in the chapter “POI Command Processors” in the *Command and Function Reference*.

Syntax:

```
QUERY
   {SCHEDULE(ACTIVE) SOURCE}
```

Returns the name of the schedule that is loaded into the ACTIVE schedule.

Example: Query the schedule name

Command (and response) to query the name of the schedule that is loaded into the ACTIVE schedule on system OPS44R:

```
OI SM$APIQ QUERY SCHEDULE(OPS44R>ACTIVE) SOURCE
SM$APIQ MESSAGE: LOAD_GET API RESPONSE (continued...)
SM$APIQ MESSAGE: OPS44R>SLEJO01>15:42:17.590471 GLOBAL0.ATMSM_ACTIVE_OPS44R=(PR
ODUCTION ON OPS44R)
SM$APIQ Result=:0:1:PRODUCTION ON OPS44R:
  1:PRODUCTION ON OPS44R
***
```

Syntax:

```
QUERY
   {SCHEDULE(*)}
```

Returns a list of all schedules that are defined in Schedule Manager.

Example: Query names of all schedules

Command (and response) to query the names of all schedules that are defined in Schedule Manager on the local system:

```
OI SM$APIQ QUERY SCHEDULE(*)
SM$APIQ Result=:0:7:DEVELOPMENT:PRODUCTION:PRODUCTIONA:PRODUCTIONB:PRODUCTIONC:S
ERVER:TESTBED:
   1:DEVELOPMENT
   2:PRODUCTION
   3:PRODUCTIONA
   4:PRODUCTIONB
   5:PRODUCTIONC
   6:SERVER
   7:TESTBED
***
```

Syntax:

```
QUERY
   {PERIOD(*)}
   {SCHEDULE(ssss)}
```

Return a list of all periods that are defined in schedule *ssss*.

Example: Query the names of all periods

Command (and response) to query the names of all periods that are defined in schedule PRODUCTION on the local system:

```
OI SM$APIQ QUERY SCHEDULE(PRODUCTION) PERIOD(*)
SM$APIQ Result=:0:13:WENDLITE S.....S 0800-2000:WENDDARK S.....S 2000-3200:EVER
YNOON SMTWTFS 1200-1300:WEEKDAY .MTWTF. 0800-1600:WEEKEVENING .MTWTF. 1600-2400:
WEEKNIGHT .MTWTF. 2400-3200:THNXLITE03 11/27/2007-11/28 0800-2000:THNXDARK03 11/
27/2
  1:WENDLITE S.....S 0800-2000
  2:WENDDARK S.....S 2000-3200
  3:EVERYNOON SMTWTFS 1200-1300
  4:WEEKDAY .MTWTF. 0800-1600
  5:WEEKEVENING .MTWTF. 1600-2400
  6:WEEKNIGHT .MTWTF. 2400-3200
  7:THNXLITE03 11/27/2007-11/28 0800-2000
  8:THNXDARK03 11/27/2007-11/28 2000-3200
  9:XMASLITE03 12/24/2007-12/25 0800-2000
 10:XMASDARK03 12/24/2007-12/25 2000-3200
 11:XMASALL03 12/25/2007 0000-2400
 12:XMASEVERY 12/25 0000-2400
 13:DEFAULT
***
```

Syntax:

```
QUERY
   {PERIOD(pppp)}
   {SCHEDULE(ssss)}
```

Return a list of all resources that are linked to period *pppp* in schedule *ssss*.

Example: Query the names of all resources

Command (and response) to query the names of all resources that are linked to period EVERYNOON in schedule PRODUCTION on the local system:

```
OI SM$APIQ QUERY SCHEDULE(PRODUCTION) PERIOD(EVERYNOON)
SM$APIQ Result=:0:1:DB2_STCTBL.ACF2:
  1:DB2_STCTBL.DB2
***
```

Syntax:

```
QUERY
   {SCHEDULE(ssss)}
   {TABLE(*)}
```

Returns a list of all SSM resource tables.

Note: Keyword SCHEDULE(*ssss*) is required to allow the user to specify the system to be queried. You must specify a valid schedule name, although only the *system>* portion (if any) of the specified schedule name affects the operation of this command.

Example: Query the name of all SSM resource tables

Command (and response) to query the name of all SSM resource tables on the local system:

```
OI SM$APIQ QUERY SCHEDULE(PRODUCTION) TABLE(*)
SM$APIQ Result=:0:6: ACF2_STCTBL: CICS_STCTBL: DB2_STCTBL: IMS_STCTB
L:JES2_STCTBL: TSO_STCTBL:
  1:ACF2_STCTBL
  2:CICS_STCTBL
  3:DB2_STCTBL
  4:IMS_STCTBL
  5:JES2_STCTBL
  6:TSO_STCTBL
***
```

Syntax:

```
QUERY
   {RESOURCE(*)}
   {SCHEDULE(ssss)}
   {TABLE(tttt)}
```

Returns a list of all resources in SSM resource table *tttt*.

Note: Keyword SCHEDULE(*ssss*) is required to allow the user to specify the system to be queried. You must specify a valid schedule name, although only the *system>* portion (if any) of the specified schedule name affects the operation of this command.

Example: Query the names of all resources in SSM resource table

Command (and response) to query the names of all resources in SSM resource table TSO_STCTBL on the local system:

```
OI SM$APIQ QUERY SCHEDULE(PRODUCTION) TABLE(TSO_STCTBL) RESOURCE(*)
SM$APIQ Result=:0:2:TSO:TSOTEST:
  1:TSO
  2:TSOTEST
***
```

Syntax:

```
QUERY
   {RESOURCE(rrrr)}
   {SCHEDULE(ssss)}
   {TABLE(tttt)}
```

Returns a list of all periods defined in schedule *ssss* that are linked to resource *rrrr* in table *tttt*.

Example: query the names of all periods

Command (and response) to query the names of all periods defined in schedule PRODUCTION on the local system that are linked to resource TSO in table TSO_STCTBL:

```
OI SM$APIQ QUERY SCHEDULE(PRODUCTION) TABLE(TSO_STCTBL) RESOURCE(TSO)
SM$APIQ Result=:0:2:WEEKDAY:XMASALL03:
  1:WEEKDAY
  2:XMASALL03
***
```

Syntax:

```
QUERY
   {PERIOD(pppp)}
   {RESOURCE(rrrr)}
   {SCHEDULE(ssss)}
   {TABLE(tttt)}
```

Returns the DESIRED_STATE of the link defined in schedule *ssss* between period *pppp* and resource *rrrr* in table *tttt*, or else return nothing if there is no link defined.

Example: query the DESIRED_STATE of the link

Command (and response) to query the DESIRED_STATE of the link defined in schedule PRODUCTION on the local system between period WEEKDAY and resource TSO in table TSO_STCTBL:

```
OI SM$APIQ QUERY SCHEDULE(PRODUCTION) TABLE(TSO_STCTBL) RESOURCE(TSO)
          PERIOD(WEEKDAY)
SM$APIQ Result=:0:1:OFF:
  1:OFF
***
```

Example: query the DESIRED_STATE of a non-existent link

Command (and response) to query the DESIRED_STATE of a non-existent link in schedule PRODUCTION on the local system between period WEEKNIGHT and resource TSO in table TSO_STCTBL:

```
OI SM$APIQ QUERY SCHEDULE(PRODUCTION) TABLE(TSO_STCTBL) RESOURCE(TSO)

       PERIOD(WEEKNIGHT)
SM$APIQ Result=:0:0:
***
```

Keywords on the Schedule Manager API QUERY Command

The following is a list of keywords on the API QUERY command and their values:

PERIOD*(period_name*)

Any valid Schedule Manager schedule period name or the wildcard character *.

RESOURCE(*resource_name*)

Any valid SSM resource name or the wildcard character *.

SCHEDULE(*schedule_name*)

Any valid Schedule Manager schedule name or the wildcard character *.

Note that the name of a schedule (including the wildcard character *) on an MSF-connected system is allowed when *schedule_name* is fully qualified as *system>name*.

TABLE*(table_name*)

Any valid SSM resource table name or the wildcard character *.

Formats of All Other Schedule Manager API Commands

Examples given in this section were generated by calling the following REXX program named SM$API to call the Schedule Manager API and display the results at the terminal:

```
/*--------------------------------------------------------------------*/
/*  REXX test ASOSMAPI API calls.                                     */
/*--+----1----+----2----+----3----+----4----+----5----+----6----+---*/
/*                                                                    */
  PARSE UPPER ARG sm_api_parms
  CALL ASOSMAPI sm_api_parms
  SAY 'SM$API Result='result
EXIT result
```

Program SM$API above was called by issuing the following command from the ISPF command shell on a TSO userid logged on to a system with a single copy of CA OPS/MVS running:

```
OI SM$API arg
```

Argument *arg* in the above command text contains the command syntax to call Schedule Manager, documented below.

For more information on the OI command, see OPSIMEX Command Processor in the chapter “POI Command Processors” in the *Command and Function Reference*.

Syntax to copy a schedule on the local system:

```
COPY
   {SCHEDULE(ssss)}
   {TARGET(tttt)}
```

Copy schedule *ssss* to schedule *tttt*. Schedule *tttt* is deleted before the copy is made so that schedule *tttt* is identical to schedule *ssss* after the copy.

Example: copy schedule on the local system

Command (and response) to copy schedule PRODUCTION on the local system to schedule TEST_IMAGE on the local system:

```
OI SM$API COPY SCHEDULE(PRODUCTION) TARGET(TEST_IMAGE)
SM$API MESSAGE: *COPIED
SM$API Result=0
***
```

Syntax to create an empty schedule:

```
Create
   {SCHEDULE(ssss)}
```

Example: Create empty schedule

Command (and response) to create empty schedule TEST_EMPTY on the local system:

```
OI SM$API CREATE SCHEDULE(TEST_EMPTY)
SM$API MESSAGE: *CREATED
SM$API Result=0
***
```

Syntax to delete a schedule on the local system:

```
DELETE
   {SCHEDULE(ssss)}
```

Example: Delete schedule on local system

Command (and response) to delete schedule TEST_IMAGE on the local system:

```
OI SM$API DELETE SCHEDULE(TEST_IMAGE)
SM$API MESSAGE: *DELETED
SM$API Result=0
***
```

Syntax to insert empty period in the schedule

```
INSERT
   {SCHEDULE(ssss)}
   {PERIOD(pppp)}
   {DATE(dddd)}
   {TIME(tttt-uuuu)}
```

Insert (create) empty period *pppp* in schedule *ssss* on the local system, active on days *dddd* with a start time of *tttt* and end time of *uuuu*.

Example: Insert empty period in schedule

Command (and response) to insert empty period WENDALL in schedule TEST_EMPTY on the local system, active all day Saturday and Sunday:

```
OI SM$API INSERT SCHEDULE(TEST_EMPTY) PERIOD(WENDALL)
                 DATE(S.....S) TIME(0000-2400)
SM$API Result=0
***
```

Syntax to create a link in the schedule

```
LINK
   {SCHEDULE(ssss)}
   {PERIOD(pppp)}
   {TABLE(tttt)}
   {RESOURCE(rrrr)}
   {DESIRED_STATE(dddd)}
```

Create a link in schedule *ssss* between period *pppp* and resource *rrrr* in table *tttt* with a DESIRED_STATE of *dddd*.

Example: Create a link in schedule

Command (and response) to create a link in schedule TEST_EMPTY on the local system between period WENDALL and resource JES2 in table JES2_STCTBL with a DESIRED_STATE of ON:

```
OI SM$API LINK SCHEDULE(TEST_EMPTY) PERIOD(WENDALL)
               TABLE(JES2_STCTBL) RESOURCE(JES2)
               DESIRED_STATE(ON)
SM$API Result=0
***
```

Syntax to create links in the schedule:

```
LINK_COPY
   {SCHEDULE(ssss)}
   {PERIOD(pppp)}
   {TARGET_PERIOD(tttt)}
```

Create links in schedule *ssss* between period *tttt* and all of the same resources that are linked to period *pppp*.

Example: Create links in schedule

Command (and response) to create links in schedule PRODUCTION on the local system between period EVERYNOON and all of the same resources that are linked to period DEFAULT:

```
OI SM$API LINKS_COPY SCHEDULE(PRODUCTION) PERIOD(DEFAULT)
                     TARGET_PERIOD(EVERYNOON)
SM$API Result=0
***
```

Syntax to create links to a schedule:

```
LINK_COPY
   {SCHEDULE(ssss)}
   {TABLE(tttt)}
   {RESOURCE(rrrr)}
   {TARGET_TABLE(aaaa)}
   {TARGET_RESOURCE(cccc)}
```

Create links in schedule *ssss* between resource *cccc* in table *aaaa* and all of the same periods that are linked to resource *rrrr* in table *tttt*.

Example: Create links to schedule

Command (and response) to create links to schedule TEST_EMPTY on the local system between resource DB2 in table DB2_STCTBL and all of the same periods that are linked to resource JES2 in table JES2_STCTBL:

```
OI SM$API LINKS_COPY SCHEDULE(TEST_EMPTY)
                     TABLE(JES2_STCTBL) RESOURCE(JES2)
                     TARGET_TABLE(DB2_STCTBL) TARGET_RESOURCE(DB2)
SM$API Result=0
***
```

Syntax to delete all links in a schedule:

```
LINK_DELETE
   {SCHEDULE(ssss)}
   {PERIOD(pppp)}
```

Delete all links in schedule *ssss* between all resources and period *pppp*.

Example: Delete all links in schedule

Command (and response) to delete all links in schedule PRODUCTION on the local system between all resources and period EVERYNOON:

```
OI SM$API LINKS_DELETE SCHEDULE(PRODUCTION) PERIOD(EVERYNOON)
SM$API Result=0
***
```

Syntax to delete all links:

```
LINK_DELETE
   {SCHEDULE(ssss)}
   {PERIOD(pppp)}
   {RESOURCE(rrrr)}
```

Delete all links in schedule *ssss* between all periods and resource *rrrr* in table *tttt*.

Example: Delete all links

Command (and response) to delete all links in schedule TEST_EMPTY on the local system between all periods and resource DB2 in table DB2_STCTBL:

```
OI SM$API LINKS_DELETE SCHEDULE(TEST_EMPTY)
                TABLE(DB2_STCTBL) RESOURCE(DB2)
SM$API Result=0
***
```

Syntax to load a schedule:

```
LOAD
   {SCHEDULE(ssss)}
   {TARGET(ACTIVE)}
```

Load (activate) schedule *ssss* by making it the ACTIVE schedule.

Example: Load schedule

Command (and response) to load (activate) schedule PRODUCTION on the local system by making it the ACTIVE schedule on the local system:

```
OI SM$API LOAD SCHEDULE(DEVELOPMENT) TARGET(ACTIVE)
SM$API MESSAGE: LOAD_SET API RESPONSE (continued...)
SM$API MESSAGE: OPS44R>SLEJO01>16:06:49.286188 GLOBAL0.ATMSM_ACTIVE_OPS44R=()
SM$API MESSAGE: LOAD_SET API RESPONSE (continued...)
SM$API MESSAGE: OPS44R>SLEJO01>16:06:49.370434
                LOBAL0.ATMSM_ACTIVE_OPS44R=(PRODUCTION ON OPS44R)
SM$API MESSAGE: *LOADED
SM$API Result=0
***
```

Syntax to merge a schedule:

```
MERGE
   {SCHEDULE(ssss)}
   {TARGET(tttt)}
```

Merge schedule *ssss* and schedule *tttt* and place the result in schedule *tttt*.

Example: Merge schedule

Command (and response) to merge schedule TEST_EMPTY on the local system and schedule PRODUCTION on the local system and place the result in schedule PRODUCTION on the local system:

```
OI SM$API MERGE SCHEDULE(TEST_EMPTY) TARGET(PRODUCTION)
SM$API MESSAGE: *MERGED
SM$API Result=0
***
```

Syntax to remove a schedule:

```
REMOVE
   {SCHEDULE(ssss)}
   {PERIOD(pppp)}
```

Remove period *pppp* from schedule *ssss*.

Example: Remove period from schedule

Command (and response) to remove period EVERYNOON from schedule PRODUCTION on the local system:

```
OI SM$API REMOVE SCHEDULE(TEST_EMPTY) PERIOD(EVERYNOON)
 SM$API Result=0
***
```

Syntax to print a report of a schedule:

```
REPORT
   {SCHEDULE(ssss)}
   [DSNAME(dddd)]
```

Print a report on links between periods and resources in schedule *ssss* to data set *dddd*, if specified, or to the default data set if *dddd* is not specified. For a description of the default data set name, see Printing Schedule Manager Data in this chapter. The report generated by this command is identical to the report generated by the Edit primary command REPORT LINKS.

If this command is successful, then it returns the name of the data set containing the report. If this command is not successful, then it returns a value of 8.

Example: Print a report

Command (and response) to print a report on the links between periods and resources in schedule PRODUCTION on the local system:

```
OI SM$API REPORT SCHEDULE(PRODUCTION) DSNAME('JOEUSER.TEST.REPORT')
SM$API Result='JOEUSER.TEST.REPORT'
***
```

The following would be the contents of data set JOEUSER.TEST.REPORT:

```
Schedule: OPS44R>PRODUCTION ----------- LINKS ------------ 15 Apr 2007

                            * * * PERIOD LIST * * *

  WENDALL     S.....S          0000-2400
  WENDLITE    S.....S          0800-2000
  WENDDARK    S.....S          2000-3200
  WEEKDAY     .MTWTF.          0800-1600
  WEEKEVENING .MTWTF.          1600-2400
  WEEKNIGHT   .MTWTF.          2400-3200
  THNXLITE03  11/27/2007-11/28 0800-2000
  THNXDARK03  11/27/2007-11/28 2000-3200
  XMASLITE03  12/24/2007-12/25 0800-2000
  XMASDARK03  12/24/2007-12/25 2000-3200
  XMASALL03   12/25/2007       0000-2400
  XMASEVERY   12/25            0000-2400
  DEFAULT
 1Schedule: OPS44R>PRODUCTION ----------- LINKS ----------- 15 Apr 2007

                           * * * SORTED BY PERIOD * * *

  Period: DEFAULT
    Table: ACF2_STCTBL
      ACF2               ON

    Table: JES2_STCTBL
      JES2               ON
  Period: THNXDARK03     11/27/2007-11/28 2000-3200
    Table: ACF2_STCTBL
      ACF2               ON

    Table: CICS_STCTBL
      CICS               OFF

    Table: IMS_STCTBL
      IMS                OFF

    Table: JES2_STCTBL
      JES2               ON

    Table: TSO_STCTBL
      TCAS               ON
```

```
 Period: THNXLITE03     11/27/2007-11/28 0800-2000
   Table: ACF2_STCTBL
     ACF2               ON

   Table: JES2_STCTBL
     JES2               ON

   Table: TSO_STCTBL
     TCAS               ON
 Period: WEEKDAY        .MTWTF.          0800-1600
   Table: ACF2_STCTBL
     ACF2               ON

   Table: CICS_STCTBL
     CICS               ON
     CICSTEST           OFF

   Table: IMS_STCTBL
     IMS                ON

   Table: JES2_STCTBL
     JES2               ON

   Table: TSO_STCTBL
     TCAS               ON
     TCASTEST           OFF
1Schedule: OPS44R>PRODUCTION ----------- LINKS ----------- 15 Apr 2007

                           * * * SORTED BY PERIOD * * *

 Period: WEEKEVENING    .MTWTF.          1600-2400
   Table: ACF2_STCTBL
     ACF2               ON

   Table: JES2_STCTBL
     JES2               ON

   Table: TSO_STCTBL
     TCAS               ON
```

```
 Period: WEEKNIGHT      .MTWTF.          2400-3200
   Table: ACF2_STCTBL
     ACF2               ON

   Table: CICS_STCTBL
     CICS               OFF

   Table: IMS_STCTBL
     IMS                OFF

   Table: JES2_STCTBL
     JES2               ON

   Table: TSO_STCTBL
     TCAS               ON
 Period: WENDALL        S.....S          0000-2400
   Table: JES2_STCTBL
     JES2               ON
 Period: WENDDARK       S.....S          2000-3200
   Table: ACF2_STCTBL
     ACF2               ON

   Table: CICS_STCTBL
     CICS               OFF

   Table: IMS_STCTBL
     IMS                OFF

   Table: JES2_STCTBL
     JES2               ON

   Table: TSO_STCTBL
     TCAS               ON
1Schedule: OPS44R>PRODUCTION ----------- LINKS ----------- 15 Apr 2007

                          * * * SORTED BY PERIOD * * *

 Period: WENDLITE       S.....S          0800-2000
   Table: ACF2_STCTBL
     ACF2               ON

   Table: JES2_STCTBL
     JES2               ON

   Table: TSO_STCTBL
     TCAS               ON
```

```
Period: XMASALL03      12/25/2007       0000-2400
  Table: DB2_STCTBL
    DB2                OFF

  Table: TSO_STCTBL
    TCASTEST           OFF
Period: XMASDARK03     12/24/2007-12/25 2000-3200
  Table: ACF2_STCTBL
    ACF2               ON

  Table: CICS_STCTBL
    CICS               OFF

  Table: IMS_STCTBL
    IMS                OFF

  Table: JES2_STCTBL
    JES2               ON

  Table: TSO_STCTBL
    TCAS               ON
Period: XMASEVERY      12/25            0000-2400
  Table: CICS_STCTBL
    CICSTEST           OFF
Period: XMASLITE03     12/24/2007-12/25 0800-2000
  Table: ACF2_STCTBL
    ACF2               ON

  Table: DB2_STCTBL
    DB2                ON

  Table: JES2_STCTBL
    JES2               ON

  Table: TSO_STCTBL
    TCAS               ON
```

```
1Schedule: OPS44R>PRODUCTION ----------- LINKS ---------- 15 Apr 2007

                          * * * SORTED BY RESOURCE * * *

  Resource: ACF2_STCTBL.ACF2
    ON  DEFAULT
    ON  THNXDARK03  11/27/2007-11/28 2000-3200
    ON  THNXLITE03  11/27/2007-11/28 0800-2000
    ON  WEEKDAY     .MTWTF.          0800-1600
    ON  WEEKEVENING .MTWTF.          1600-2400
    ON  WEEKNIGHT   .MTWTF.          2400-3200
    ON  WENDDARK    S.....S          2000-3200
    ON  WENDLITE    S.....S          0800-2000
    ON  XMASDARK03  12/24/2007-12/25 2000-3200
    ON  XMASLITE03  12/24/2007-12/25 0800-2000
  Resource: CICS_STCTBL.CICS
    OFF THNXDARK03  11/27/2007-11/28 2000-3200
    ON  WEEKDAY     .MTWTF.          0800-1600
    OFF WEEKNIGHT   .MTWTF.          2400-3200
    OFF WENDDARK    S.....S          2000-3200
    OFF XMASDARK03  12/24/2007-12/25 2000-3200
  Resource: CICS_STCTBL.CICSTEST
    OFF WEEKDAY     .MTWTF.          0800-1600
    OFF XMASEVERY   12/25            0000-2400

Resource: DB2_STCTBL.DB2
    ON  XMASLITE03  12/24/2007-12/25 0800-2000
    OFF XMASALL03   12/25/2007       0000-2400

Resource: IMS_STCTBL.IMS
    OFF THNXDARK03  11/27/2007-11/28 2000-3200
    ON  WEEKDAY     .MTWTF.          0800-1600
    OFF WEEKNIGHT   .MTWTF.          2400-3200
    OFF WENDDARK    S.....S          2000-3200
    OFF XMASDARK03  12/24/2007-12/25 2000-3200

  Resource: JES2_STCTBL.JES2
    ON  DEFAULT
    ON  THNXDARK03  11/27/2007-11/28 2000-3200
    ON  THNXLITE03  11/27/2007-11/28 0800-2000
    ON  WEEKDAY     .MTWTF.          0800-1600
    ON  WEEKEVENING .MTWTF.          1600-2400
    ON  WEEKNIGHT   .MTWTF.          2400-3200
    ON  WENDALL     S.....S          0000-2400
    ON  WENDDARK    S.....S          2000-3200
    ON  WENDLITE    S.....S          0800-2000
    ON  XMASDARK03  12/24/2007-12/25 2000-3200
    ON  XMASLITE03  12/24/2007-12/25 0800-2000
```

```
1Schedule: OPS44R>PRODUCTION ----------- LINKS ------------ 15 Apr 2007

                          * * * SORTED BY RESOURCE * * *

  Resource: TSO_STCTBL.TCAS
    ON  THNXDARK03  11/27/2007-11/28 2000-3200
    ON  THNXLITE03  11/27/2007-11/28 0800-2000
    ON  WEEKDAY     .MTWTF.          0800-1600
    ON  WEEKEVENING .MTWTF.          1600-2400
    ON  WEEKNIGHT   .MTWTF.          2400-3200
    ON  WENDDARK    S.....S          2000-3200
    ON  WENDLITE    S.....S          0800-2000
    ON  XMASDARK03  12/24/2007-12/25 2000-3200
    ON  XMASLITE03  12/24/2007-12/25 0800-2000

  Resource: TSO_STCTBL.TCASTEST
    OFF WEEKDAY     .MTWTF.          0800-1600
    OFF XMASALL03   12/25/2007       0000-2400
```

Syntax to have Schedule Manager perform a reset function:

```
RESET OI
    [SCHEDULE(ACTIVE)]
```

Request Schedule Manager to perform a RESET function resulting in the following:

- The ACTIVE schedule and the @OVERRIDE@ schedule are reevaluated.
- The desired states of SSM resources are set as scheduled.
- The next schedule RESET time (period start or stop) is recomputed.

The Schedule Manager RESET function is always executed using the ACTIVE schedule and the @OVERRIDE@ schedule, if one exists. The optional keyword SCHEDULE may be specified on the RESET_OI command to direct the request to another system. If the keyword SCHEDULE is specified, then the schedule name must be ACTIVE. If the keyword SCHEDULE is not specified, then the request is directed to the local system.

Example: Have Schedule Manager perform a RESET function

Command (and response) to have Schedule Manager perform a RESET function on the local system:

```
OI SM$API RESET_OI
SM$API MESSAGE: RESET SCHEDULED (continued...)
SM$API MESSAGE: Check OSF logs on system OPS44R.
SM$API Result=0
***
```

Syntax to synchronize a schedule:

```
SYNCH_AUTO
    {SCHEDULE(ssss)}
```

Synchronize schedule *ssss* with SSM table updates by automatically removing all links containing deleted SSM tables, resources, or both.

This command and the Y (synch) line command in Edit cannot be executed against the ACTIVE schedule. If the ACTIVE schedule becomes out-of-synch due to deleted SSM tables, resources or both, then the source schedule used to build the ACTIVE schedule will also be out-of-synch. You can use the SYNCH_AUTO command to synchronize the source schedule and then use the LOAD command to re-load it into the ACTIVE schedule.

Schedule Manager issues the following WTO during its RESET processing whenever it recognizes the ACTIVE schedule as being out-of-synch:

OPS7913E FATAL|Non-fatal SSM Schedule Manager error. ACTIVE schedule out-of-synch. Synch and re-load source schedule *sssssssssss.*

Schedule *sssssssssss* last loaded into the ACTIVE schedule is listed in the message text. It is most likely out-of-synch also. The error condition is not resolved until a synchronized schedule is loaded into the ACTIVE schedule.

If period definitions are missing, then this is a FATAL error and it will be indicated in the WTO text.

Notes:

- Schedule Manager RESET processing halts after encountering a fatal error due to missing period definitions and does not restart until the error condition is resolved. In other words, Schedule Manager no longer schedules the desired states of your SSM resources.
- A fatal error due to missing period definitions cannot be resolved programmatically using the Schedule Manager API. The user has to go into the ISPF Schedule Manager application (OPSVIEW option 4.11.4) to resolve the errors.

Example: Synchronize schedule

Failing command (and response) to synchronize schedule PRODUCTION on the local system that contains links to deleted period definitions:

```
OI SM$API SYNCH_AUTO SCHEDULE(DAMAGED)
SM$API MESSAGE: MISSING PERIODS (continued...)
SM$API MESSAGE: Links to missing SSM tables/resources were deleted. Use OPSVIEW
                4.11.4 to SYNCH other links to missing period definitions.
SM$API Result=8
***
```

If SSM tables, resources, or both are missing, then this is a non-fatal error and it will be indicated in the WTO text. Schedule Manager RESET processing continues for remaining SSM resources after encountering a non-fatal error. In other words, Schedule Manager continues to schedule the desired states of your remaining SSM resources. A non-fatal error due to missing SSM tables, resources, or both can be resolved programmatically using the Schedule Manager API commands SYNCH_AUTO and LOAD, as described above. Of course, the user can also resolve the errors using the ISPF Schedule Manager application (OPSVIEW option 4.11.4).

Note: An AOF MSG rule could be written against the OPS7913E WTO to automate a programmatic resolution to a non-fatal Schedule Manager error due to an out-of-synch ACTIVE schedule.

Example: Remove all links to deleted SSM tables

Command (and response) to remove all links to deleted SSM tables, resources, or both from all periods defined in schedule PRODUCTION on the local system:

```
OI SM$API SYNCH_AUTO SCHEDULE(PRODUCTION)
SM$API MESSAGE: *SYNCHED
SM$API Result=0
***
```

Syntax to delete a single link:

```
UNLINK
   {SCHEDULE(ssss)}
   {PERIOD(pppp)}
   {TABLE(tttt)}
   {RESOURCE(rrrr)}
```

Delete single link in schedule *ssss* between period *pppp* and resource *rrrr* in table *tttt*.

Example: Delete link

Command (and response) to delete the link in schedule PRODUCTION on the local system between period DEFAULT and resource ACF2 in table ACF2_STCTBL:

```
OI SM$API UNLINK SCHEDULE(PRODUCTION) PERIOD(DEFAULT)
              TABLE(ACF2_STCTBL) RESOURCE(ACF2)
SM$API Result=0
***
```

Keywords on All Other Schedule Manager API Commands

The following is a list of keywords on all other Schedule Manager API commands and their values:

DATE(*date*)

mask

Defines a cyclic days-of-week period where *mask* is a seven-character mask of the format SMTWTFS specifying the weekdays that the period is active, with a dot indicating each weekday that the period is inactive. For example, the mask SM…FS specifies a period that is active on Sunday, Monday, Friday, and Saturday.

***mm/dd*[*/yyyy*]**

Defines a fixed-date period active on one day *dd* of one month *mm* only. If optional year *yyyy* is specified, then the period is active in year *yyyy* only. Otherwise, the period is active on day *mm/dd* of every year (unless day 02/29 is specified, in which case the period is active only in leap years).

mm/dd*[*/yyyy*] *-nn/ee

Definnes a fixed-date period active during a range of dates, starting on day *dd* of month *mm* and ending on day *ee* of month *nn*. If optional year *yyyy* is specified, then the period starts in year *yyyy* only. Otherwise, the period starts on day *mm/dd* of every year (unless day 02/29 is specified, in which case the period starts only in leap years).

DESIRED_STATE(*desired_state*)

ON|OFF

DSNAME(*data_set_name*)

Any valid TSO data set name.

PERIOD(*period_name*)

Any valid Schedule Manager schedule period name.

RESOURCE(*resource_name*)

Any valid SSM resource name.

SCHEDULE(*schedule_name*)

Any valid Schedule Manger schedule name. Note that the *name* of a schedule on an MSF-connected system is allowed when *schedule_name* is fully qualified as *system>name*.

TABLE(*table_name*)

Any valid SSM resource table name.

TARGET(*tschedule_name*)

Any valid Schedule Manager schedule name.

Note that the *name* of a schedule on an MSF-connected system is allowed when *schedule_name* is fully qualified as *system>name*.

TARGET_PERIOD(*period_name*)

Any valid Schedule Manager schedule period name.

TARGET_RESOURCE(*resource_name*)

Any valid SSM resource name.

TARGET_TABLE(*table_name*)

Any valid SSM resource table name.

TIME(*time*)

hhmm-iinn-The period start time *hhmm* and the period stop time *iinn* on a 48-hour military clock. Stop time *iinn* must be greater than start time *hhmm*.

Override Schedule Manager

This section discusses the various ways you can override Schedule Manager.

The Effective Mode of SSM Resources

The effective mode of any resource is determined by the most restrictive of the following modes:

- The global mode of System State Manager.

  This mode is set by the CA OPS/MVS parameter STATEMAN.

- The mode of the resource table that contains the resource.

  This mode is set by the value for the resource table in the TABLE_MODE column in the SSM_MANAGED_TBLS table.

- The mode of the individual resource.

  This mode is set by the value for the resource in the MODE column in the resource table.

The CA OPS/MVS parameter STATESCHEDEXCLUDE may be used to have Schedule Manager bypass updates to the DESIRED_STATE column of resources during its RESET processing based on the effective mode of each resource. For more information about this parameter, see the *Parameter Reference*.

System State Manager Resource Tables

Schedule Manager RESET processing looks for a SCHEDMODE column in a resource table. If one is present, then Schedule Manager looks for a value of INACTIVE. For any resource having a value of INACTIVE in its SCHEDMODE column, Schedule Manager bypasses any updates to its DESIRED_STATE column during RESET processing.

To use this feature, manually add a SCHEDMODE column to resource tables if one is not already present. The BASIC and STC model tables available in the RDF Table Editor (OPSVIEW primary option 2.6) contain a SCHEDMODE column with a default value of ACTIVE.

This feature makes it possible for the user to override the scheduled state of a resource through a single SQL statement similar to the following example:

```
UPDATE table SET SCHEDMODE='INACTIVE' WHERE NAME='resource'
```

Likewise, the user can return control of the *resource* to Schedule Manager through a single SQL statement similar to the following example:

```
UPDATE table SET SCHEDMODE='ACTIVE' WHERE NAME='resource'
```

Such SQL statements can be inserted into user-coded CA OPS/MVS rules to override the scheduled state of selected resources while allowing Schedule Manager to maintain the usual scheduled states of other system resources. For example, the user can code emergency START or STOP started task command (CMD) rules that set the SCHEDMODE of the resource to INACTIVE and the DESIRED_STATE of the resource to UP or DOWN.

SSMSCHED Sample Rule

The SSMSCHED sample command rule allows operators or automated procedures to set or display the SCHEDMODE of resources. You must enable the SSMSCHED sample rule before you can begin to use it. Descriptions of the two SSMSCHED commands follow.

Syntax to update the SSM schedule:

```
SSMSCHED UPDATE
   {RESOURCE(table.resource, . . .)}
   {SCHEDMODE(ACTIVE|INACTIVE)}
```

This command sets the SCHEDMODE values for each *table.resource* entry specified in the RESOURCE() list to the value specified on the SCHEDMODE() keyword. For each *table.resource* entry specified in the RESOURCE() list, the following SQL statement is executed:

```
UPDATE table SET SCHEDMODE='mode' WHERE NAME LIKE 'resource'
```

As shown in the above SQL statement, each *table* value specified must be a literal value; each *resource* value specified could be any valid value specified on an SQL LIKE keyword. Specify up to twenty *table.resource* entries in the RESOURCE() list.

Examples:

1. Command (and response) to turn off Schedule Manger control of some resources:

```
SSMSCHED UPDATE RESOURCE(CICS_STCTBL.CICS%) SCHEDMODE(INACTIVE)
OPR1000I SSMSCHED: UPDATE returned RC=00 / SQLCODE=00000 for resource
CICS_STCTBL.CICS%
```

2. Command (and response) to turn on Schedule Manager control of a resource:

```
SSMSCHED UPDATE RESOURCE(CICS_STCTBL.CICSTEST) SCHEDMODE(ACTIVE)
OPR1000I SSMSCHED: UPDATE returned RC=00 / SQLCODE=00000 for resource
CICS_STCTBL.CICSTEST
```

Syntax to select the SSM schedule:

```
SSMSCHED SELECT
   {RESOURCE(table.resource,...)}
   [SCHEDMODE(ACTIVE|INACTIVE)]
```

This command displays the SCHEDMODE values for each *table.resource* entry specified in the RESOURCE() list. Optionally, you may limit the display to only those resources whose SCHEDMODE value is equal to the value specified on the SCHEDMODE() keyword. For each *table.resource* entry specified in the RESOURCE() list, the following SQL statement is executed, and the results displayed:

```
SELECT SCHEDMODE NAME INTO :dmode dname FROM table
       WHERE NAME LIKE 'resource' {AND SCHEDMODE='mode'}
```

As shown in the previous SQL statement, each *table* value specified must be a literal value; each *resource* value specified could be any valid value specified on an SQL LIKE keyword. Specify up to twenty *table.resource* entries in the RESOURCE() list.

Examples:

1. Command (and response) to display the status of Schedule Manager control of some resources:

```
SSMSCHED SELECT RESOURCE(CICS_STCTBL.CICS%)
OPR1000I SSMSCHED: SELECT returned RC=00 / SQLCODE=00000 for resource
CICS_STCTBL.CICS%
OPR1000I SSMSCHED:  SCHEDMODE=INACTIVE for resource            CICS_STCTBL.CICS
OPR1000I SSMSCHED:   SCHEDMODE=ACTIVE   for resource
CICS_STCTBL.CICSTEST
```

2. Command (and response) to display only those resources in a table that are under Schedule Manager control:

```
SSMSCHED SELECT RESOURCE(CICS_STCTBL.CICS%) SCHEDMODE(ACTIVE)
OPR1000I SSMSCHED: SELECT returned RC=00 / SQLCODE=00000 for resource
CICS_STCTBL.CICS
OPR1000I SSMSCHED: SCHEDMODE=ACTIVE for resource
CICS_STCTBL.CICSTEST
```

Schedule Manager Overrides

You can schedule temporary overrides to all Schedule Manager schedules by creating a schedule named @OVERRIDE@, which is a reserved name. The @OVERRIDE@ schedule is automatically merged into all other schedules when they are edited, and it is automatically merged into the ACTIVE schedule during RESET processing to set the desired states of resources.

The period names in the @OVERRIDE@ schedule are also reserved and are prefixed with the characters @OVER. You can only make changes to the @OVERRIDE@ schedule when editing it directly. If you try to modify @OVERRIDE@ period or link entries when editing another schedule, then your changes are rejected.

The following Links Control panel shows an example of an @OVERRIDE@ schedule to turn off all resources except JES2 for one hour every night at midnight. In this example, an L line command has been executed against the @OVER_MNIT period to display its links:

```
Schedule Manager ---------------- Links Control ----------------- System:OPS44R
Command ===>                                                    Scroll ===> CSR

CMDS: A D I L LC LD LX + ++ - --              |
---------------------------------------------- -------------------------------
___ -@OVERRIDE@                                |     -ACF2_STCTBL
L__  |-@OVER_MNIT  SMTWTFS          0000-0100 |      |-STASK
___  |-DEFAULT                                | OFF  | |-ACF2
                                              |     -CICS_STCTBL
                                              |      |-STASK
                                              | OFF  | |-CICS
                                              | OFF  | |-CICSTEST
                                              |     -DB2_STCTBL
                                              |      |-STASK
                                              |      | |-DB2
                                              |     -IMS_STCTBL
                                              |      |-STASK
                                              | OFF  | |-IMS
                                              |     -JES2_STCTBL
                                              |      |-STASK
                                              |      | |-JES2
                                              |     -TSO_STCTBL
                                              |      |-STASK
                                              | OFF  | |-TCAS
                                              | OFF  | |-TCASTEST
                                              |
```

The following Links Control panel demonstrates the @OVERRIDE@ schedule being merged into the PRODUCTION schedule:

```
Schedule Manager ---------------- Links Control ----------------- System:OPS44R
Command ===>                                                      Scroll ===> CSR

CMDS: A D I L LC LD LX + ++ - --              |
---------------------------------------------- -------------------------------
___ -PRODUCTION                                |     -ACF2_STCTBL
___  |-@OVER_MNIT  SMTWTFS          0000-0100 |      |-STASK
___  |-WENDLITE    S.....S          0800-2000 |      | |-ACF2
___  |-WENDDARK    S.....S          2000-3200 |     -CICS_STCTBL
___  |-EVERYNOON   SMTWTFS          1200-1300 |      |-STASK
___  |-WEEKDAY     .MTWTF.          0800-1600 |      | |-CICS
___  |-WEEKEVENING .MTWTF.          1600-2400 |      | |-CICSTEST
___  |-WEEKNIGHT   .MTWTF.          2400-3200 |     -DB2_STCTBL
___  |-THNXLITE03  11/27/2007-11/28 0800-2000 |      |-STASK
___  |-THNXDARK03  11/27/2007-11/28 2000-3200 |      | |-DB2
___  |-XMASLITE03  12/24/2007-12/25 0800-2000 |     -IMS_STCTBL
___  |-XMASDARK03  12/24/2007-12/25 2000-3200 |      |-STASK
___  |-XMASALL03   12/25/2007       0000-2400 |      | |-IMS
___  |-XMASEVERY   12/25            0000-2400 |     -JES2_STCTBL
___  |-DEFAULT                                |      |-STASK
                                               |      | |-JES2
                                               |     -TSO_STCTBL
                                               |      |-STASK
                                               |      | |-TCAS
                                               |      | |-TCASTEST
```

The following State panel demonstrates the @OVERRIDE@ schedule being merged into the PRODUCTION schedule:

```
Schedule Manager ----- State at 0030 on 03/31/2007 (MON) -------- System:OPS44R
Command ===>                                                      Scroll ===> CSR

                                                | CMDS: F N R + ++ - --
------------------------------------------------ -------------------------------
                                                - ___ -ACF2_STCTBL
                                                - ___  |-STASK
    @OVER_MNIT     SMTWTFS          0000-0100 - OFF  | |-ACF2
                                                - ___ -CICS_STCTBL
                                                - ___  |-STASK
    @OVER_MNIT     SMTWTFS          0000-0100 - OFF  | |-CICS
    @OVER_MNIT     SMTWTFS          0000-0100 - OFF  | |-CICSTEST
                                                - ___ -DB2_STCTBL
                                                - ___  |-STASK
                                                - ___  | |-DB2
                                                - ___ -IMS_STCTBL
                                                - ___  |-STASK
    @OVER_MNIT     SMTWTFS          0000-0100 - OFF  | |-IMS
                                                - ___ -JES2_STCTBL
                                                - ___  |-STASK
    WEEKNIGHT      .MTWTF.          2400-3200 - ON_  | |-JES2
                                                - ___ -TSO_STCTBL
                                                - ___  |-STASK
    @OVER_MNIT     SMTWTFS          0000-0100 - OFF  | |-TCAS
    @OVER_MNIT     SMTWTFS          0000-0100 - OFF  | |-TCASTEST
```

The user deactivates the @OVERRIDE@ schedule by deleting it or renaming it.

Suppose, for example, that the sample @OVERRIDE@ schedule above was copied to a schedule named @OVERMNIGHT. Schedule @OVERMNIGHT could then be activated as the override schedule at any time by copying it or renaming it to schedule @OVERRIDE@.

SSM@OVER Sample Rule

The SSM@OVER sample command rule allows operators or automated procedures to maintain temporary fixed date periods in the @OVERRIDE@ schedule in the CA OPS/MVS Schedule Manager facility. This rule allows you to add, modify, display, or delete links in the @OVERRIDE@ schedule. You must enable the SSM@OVER sample rule before you can begin using it.

SSM@OVER Command Formats

The SSM@OVER commands and their formats are described below. Keywords are described in the following section.

- The SSM@OVER ADD command has the following syntax:

```
SSM@OVER ADD
   {NAMES(table.resource,...)}
   {STATE(ON|OFF|UP|DOWN|ACTIVE|INACTIVE)}
   {FROM(hhmm)}
   {UNTIL(hhmm)}
   [DATE(mm/dd)]
   [RESET]
```

  Add links with the specified STATE to the specified resource NAMES in the @OVERRIDE@ schedule. Add the links in a temporary fixed date period with the specified DATE, the specified FROM start time, and the specified UNTIL end time. If DATE is not specified, then it defaults to the current date. After the links are added, perform a Schedule Manager RESET function if RESET was specified.

  Examples

  1. Command (and response) to add some temporary override links to turn some resources ON:

```
SSM@OVER ADD NAMES(CICS_STCTBL.CICS,TSO_STCTBL.TCAS) STATE(ON) FROM(2200)
UNTIL(2400)
OPR1000I PROD.SSM@OVER: Override for CICS_STCTBL.CICS set ON at 2200-2400 03/31-03/31
added to @OVER000001.
OPR1000I PROD.SSM@OVER: Override for TSO_STCTBL.TCAS set ON at 2200-2400 03/31-03/31
added to @OVER000001.
OPR1000I SSM@OVER: Command completed with RC=0.
```

  2. Command (and response) to add some temporary override links to turn some resources OFF:

```
SSM@OVER ADD NAMES(CICS_STCTBL.CICSTEST, TSO_STCTBL.TCASTEST) STATE(OFF) FROM(2200)
UNTIL(2400)
OPR1000I SSMQAN.SSM@OVER: Override for CICS_STCTBL.CICSTEST set OFF at 2200-2400
03/31-03/31 added to @OVER000001.
OPR1000I SSMQAN.SSM@OVER: Override for TSO_STCTBL.TCASTEST set OFF at 2200-2400
03/31-03/31 added to @OVER000001.
OPR1000I SSM@OVER: Command completed with RC=0.
```

- The f SSM@OVER CHANGE command has the following syntax:

```
SSM@OVER CHANGE
   {NAMES(table.resource,...)}
   {STATE(ON|OFF|UP|DOWN|ACTIVE|INACTIVE)}
   {FROM(hhmm)}
   {UNTIL(hhmm)}
   [DATE(mm/dd)]
   [RESET]
```

  - Change existing links to the specified resource NAMES in the @OVERRIDE@ schedule. Change the links to the specified STATE and move them to a temporary fixed date period with the specified DATE, the specified FROM start time, and the specified UNTIL end time. If DATE is not specified, then it defaults to the current date. After the links are changed, perform a Schedule Manager RESET function if RESET was specified.

  - **Example 1:** Command (and response) to change a temporary override link from ON to OFF and move it to a different time on a different day:

```
SSM@OVER CHANGE NAMES(TSO_STCTBL.TCAS) STATE(OFF) FROM(2100) UNTIL(2300)
DATE(4/1)
OPR1000I PROD.SSM@OVER: Overrides for TSO_STCTBL.TCAS deleted from @OVER000001.
OPR1000I PROD.SSM@OVER: Override for TSO_STCTBL.TCAS set OFF at 2100-2300
04/01-04/01 added to @OVER000002.
OPR1000I SSM@OVER: Command completed with RC=0.
```

  - **Example2:** Command (and response) to change a temporary override link from OFF to ON and move it to a different time on a different day:

```
SSM@OVER CHANGE NAMES(TSO_STCTBL.TCASTEST) STATE(ON) FROM(2100) UNTIL(2300)
DATE(4/1)
OPR1000I PROD.SSM@OVER: Overrides for TSO_STCTBL.TCASTEST deleted from
@OVER000001.
OPR1000I PROD.SSM@OVER: Override for TSO_STCTBL.TCASTEST set ON at 2100-2300
04/01-04/01 added to @OVER000002.
OPR1000I SSM@OVER: Command completed with RC=0.
```

- The SSM@OVER DELETE command has the following syntax:

```
SSM@OVER DELETE
   {NAMES(table.resource,...)}
   [STATE(ON|OFF|UP|DOWN|ACTIVE|INACTIVE)]
   [FROM(hhmm)]
   [UNTIL(hhmm)]
   [DATE(mm/dd)]
   [RESET]
```

Delete existing links to the specified resource NAMES in the @OVERRIDE@ schedule. Optionally, delete only links with the specified STATE, and only those in the temporary fixed date period with the specified DATE, the specified FROM start time, and the specified UNTIL end time.

Example: Command (and response) to delete all temporary override links in an SSM resource table:

```
SSM@OVER DELETE NAMES(CICS_STCTBL.*)
OPR1000I PROD.SSM@OVER: Overrides for CICS_STCTBL.* deleted from @OVER000001.
OPR1000I SSM@OVER: Command completed with RC=0.
```

- The SSM@OVER RESET command has the following syntax:

  ```
  SSM@OVER RESET
  ```

  This command causes the main Schedule Manager to perform a RESET function, resulting in the following:

  - The ACTIVE schedule and the @OVERRIDE@ schedule are reevaluated.
  - The desired states of System State Manager resources are set as scheduled.
  - The next scheduled RESET time (period start or stop) is recomputed.

  Example: Command (and response) to have Schedule Manager execute a RESET function:

  ```
  SSM@OVER RESET
  OPR1000I SSM@OVER: Command completed with RC=0.
  ```

- The SSM@OVER SHOW command has the following syntax:

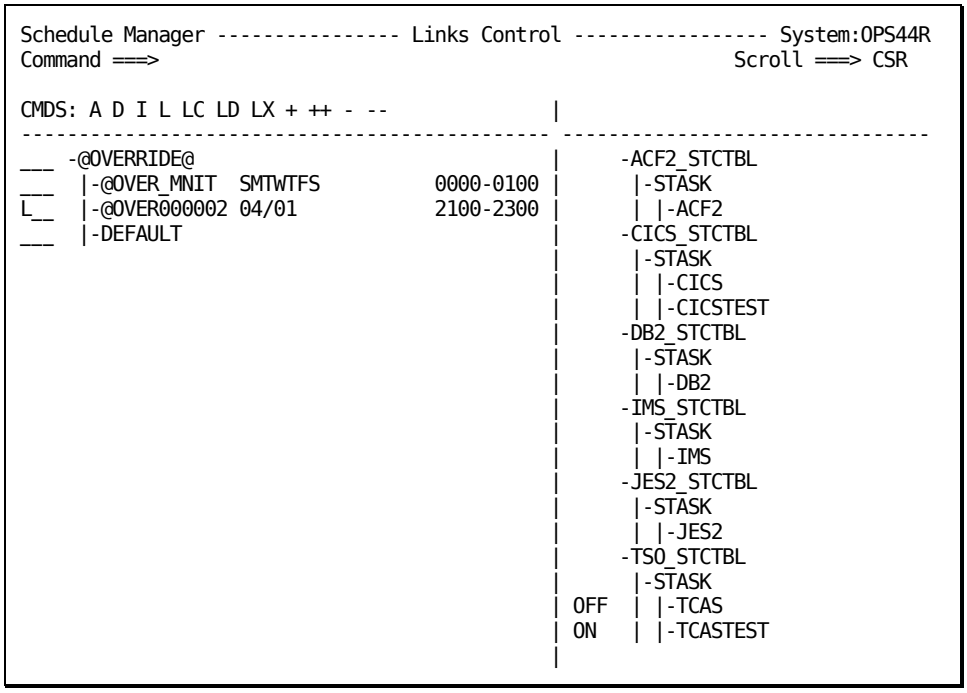
  ```
  SSM@OVER SHOW
     {NAMES(table.resource,...)}
     [STATE(ON|OFF|UP|DOWN|ACTIVE|INACTIVE)]
     [FROM(hhmm)]
     [UNTIL(hhmm)]
     [DATE(mm/dd)]
     [RESET]
  ```

  Show existing links to the specified resource NAMES in the @OVERRIDE@ schedule. Optionally, show only links with the specified STATE, and only those in the temporary fixed date period with the specified DATE, the specified FROM start time, and the specified UNTIL end time. After the links are shown, perform a Schedule Manager RESET function if RESET was specified.

  Example: Command (and response) to show all temporary override links:

  ```
  SSM@OVER SHOW
  OPR1000I PROD.SSM@OVER: TSO_STCTBL.TCAS set OFF at 2100-2300 04/01-04/01 in
  @OVER000002.
  OPR1000I PROD.SSM@OVER: TSO_STCTBL.TCASTEST set ON at 2100-2300 04/01-04/01 in
  @OVER000002.
  OPR1000I SSM@OVER: Command completed with RC=0.
  ```

The following Links Control panel shows the same example @OVERRIDE@ schedule shown earlier after all of the above example SSM@OVER commands have been executed. In this example, an L line command has been executed against the @OVER000002 period to display its links:



Keywords for the SSM@OVER Commands

The keywords for the SSM@OVER commands are:

NAMES

The list of System State Manager resource names to be acted upon. You may specify up to 20 names. Be sure to separate multiple names with a comma.

Notes:

- If the table name of the resource is not specified, the first resource table specified in the current System State Manager resource directory table that contains the specified resource name is used.
- An asterisk (*) at the end of the resource name causes all like-named resources in the table to be selected.
- An asterisk (*) at the end of the table name is permitted only for the SSM@OVER DELETE and SSM@OVER SHOW commands to process any matching table name.

STATE

The desired state for the resource during the period. Values are:

- ON or OFF
- UP or DOWN
- ACTIVE or INACTIVE

FROM

The period start time in *hhmm* format. You can use the following abbreviations for *hhmm*:

- hh
- h
- hmm

UNTIL

The period end time in *hhmm* format. You can use the following abbreviations for *hhmm*:

- hh
- h
- hmm

DATE

The period start date in *mm/dd* format. You can use the following abbreviations for *mm/dd* (zeros are added accordingly):

- m
- m/d
- m/
- /d

Notes:

If the month or day is not specified, then the current month or day is used.

This keyword defaults to the current date when the SSM@OVER ADD and SSM@OVER CHANGE commands are specified.

RESET

When specified by itself or with another command, this keyword invokes the scheduler reset function, performing desired state changes according to the active schedule and any override schedule entries.

You must use the RESET keyword when the SSM@OVER ADD, SSM@OVER CHANGE, or SSM@OVER DELETE commands have made one or more changes to the override schedule.

The reset function is not performed automatically after every override change, allowing for multiple changes and verification prior to the activation of schedule changes.

Obsolete Override Periods

Obsolete override periods created by the SSM@OVER command rule are automatically deleted whenever Schedule Manager executes a RESET function. An obsolete override period is defined as when the ending date of the period is greater than 2 and less than 60 days prior to the current date. Therefore, future overrides may be predefined up to 305 days in advance.

Chapter 11: Using the Relational Data Framework

This section contains the following topics:

The Relational Data Framework

The SQL-based Relational Data Framework provided by CA OPS/MVS stores system information used by AOF rules and automation procedures.

The CA OPS/MVS Relational Data Framework facility lets you use Structured Query Language (SQL) statements to manage the large amounts of data required by AOF rules and automation procedures. Instead of using large sets of variables, you can use the Relational Data Framework to:

- Collect data
- Organize the data into a relational table containing rows and columns of related information
- Retrieve related data by selecting it from a particular row or column
- Update data in relational tables

For reference information, see the chapter “Relational Data Framework Reference” in the Command *and Function Reference*.

Why We Chose SQL

We chose SQL to manage automation data because of the wide popularity of SQL with mainframe and PC users. The Relational Data Framework consists of relational SQL tables plus a subset of the SQL language that conforms to American National Standards Institute (ANSI) standards. If you already know SQL, you can use the CA OPS/MVS subset of it right away.

Assumptions

This guide does not attempt to completely explain SQL or relational database concepts; it assumes that CA OPS/MVS SQL users have previous SQL experience. If you need more information, consult a book about SQL. One place to start is *Understanding the New SQL: A Complete Guide* by Jim Melton and Alan R. Simon (Morgan Kaufmann Publishers).

The Role of Relational Tables

The CA OPS/MVS product uses SQL to create a database containing relational tables that store automation data. Each table contains one or more rows consisting of one or more columns. The relationship among the tables, rows, and columns of a relational table resembles the relationship among z/OS data set components, as shown in the following table:

z/OS Data Set Component	Relational Data Framework Equivalent
Data set	Relational table
Record	Row
Record field	Column

In most languages, you map the fields in a record yourself, so you know how the data is stored and which offset contains a certain field. SQL, however, controls where data goes in a table. Therefore, with SQL you do not need to know the arrangement of the data, only the names of the table and the columns where the data is located.

How the Product Stores Table Data

CA OPS/MVS implements its relational tables as a set of global variables. One global variable stores one row of a relational table, which you can access only through SQL statements.

The CA OPS/MVS product keeps Relational Data Framework global variables in a data-in-virtual (DIV) data set. Storing the variables in this way speeds up processing and allows the Relational Data Framework to operate in a time-sensitive environment. To have a permanent database for these variables, you must include the SYSCHK1 DD statement in your CA OPS/MVS startup procedure.

When CA OPS/MVS updates a row in a relational table, the change takes place in main storage, and occurs quickly. To ensure that the updated information remains safe if the system fails without warning, the representation of the relational table in main storage is periodically flushed to the DASD image of the DIV data set. However, updates that have not been saved to DASD can be lost if the system fails. By default, the flush to DASD occurs every 15 seconds (the system administrator can adjust this interval), so data that has changed in 15 seconds of failure may be lost.

When CA OPS/MVS starts up, it loads all relational tables into memory and checks the integrity of data in the tables. If any data is damaged, CA OPS/MVS either repairs the damage or marks the affected table as inaccessible.

Tables the Product Provides

Since it is so important to know the default values in your tables and the names of the tables and columns in which your data resides, CA OPS/MVS provides three tables that describe all CA OPS/MVS relational tables:

- The table named TABLE contains the names of relational tables.
- The table named COLUMN defines the columns in each table.
- The table named DEFAULT contains one row for each table that has default values; the row contains all of the default values for the table.

You cannot search, delete, or alter the DEFAULT table. The TABLE and COLUMN tables are data dictionary tables. You can search them, but you cannot delete or alter them. When you create or alter any other table using the SQL statements described in this chapter, an entry is made by CA OPS/MVS in the TABLE table for each new table, the COLUMN table for each new column, and the DEFAULT table for each new default value (if any were defined).

Table Restrictions

The CA OPS/MVS product supports up to 1000 relational tables. You can define as many as 100 columns per table, with as many rows as you need. The sum of the width of all columns cannot exceed 16,000 per table.

Reserved Keywords in SQL Statements

Following is a list of reserved keywords in SQL statements. When specifying columns for relational tables, do not use these keywords as column names. The list may grow in future versions of the SQL standard component.

ADD	DELETE	KEY	SCHEMA*
ALTER	DESC	LEADING	SECOND
AND	DISTINCT	LEFT*	SELECT
AS	DOUBLE*	LIKE	SET
ASC	DROP	LOCK*	SMALLINT
AVG	ESCAPE	LOWER	SQLCODE
BETWEEN	EXISTS*	MAX	SQLMSG
BOTH	EXTRACT	MIN	SUBSTR
BY	FETCH	MINUS*	SUM

BYTE	FIRST	MINUTE	SYNONYM*
BYTES	FLOAT*	MONTH	TABLE
CASE	FOR	NATURAL*	TEMPORARY
CHAR	FORMAT	NEXT	TIME
CHAR_LENGTH	FROM	NOT	TIMESTAMP
CLOSE	FULL*	NULL	TO
COALESCE	GLOBAL	NUMBER	TRAILING
COLUMN	GRANT*	NUMERIC	TRIGGER*
COMMIT*	GRAPHIC*	OF	TRIM
COUNT	GROUP	ON*	UNION*
CREATE	HAVING	OPEN	UNIQUE
CROSS	HEX	OPTION*	UPDATE
CURRENT	HOUR	OR	UPPER
CURSOR	IN	ORDER	USER
DATABASE	INDEX	OUTER*	USING*
DATE	INDICATOR	POSITION	VALIDATE*
DAY	INNER*	PRECISION*	VALUES
DEBUG	INSERT	PRIMARY	VARCHAR
DEC	INT	PRIOR*	VIEW*
DECIMAL	INTEGER	REAL*	WHERE
DECLARE	INTO	RIGHT*	WHILE
DEFAULT	IS	ROLLBACK*	WITH
DEFINITION	JOIN	ROW	WITHIN
			YEAR

* Currently unsupported under CA OPS/MVS. However, these keywords may be supported in future releases of the product.

Operations Performed With the CA OPS/MVS SQL

Using the CA OPS/MVS version of SQL, you can perform tasks that create, modify, access, or delete relational tables. The CA OPS/MVS SQL allows you to perform three types of operations:

- Searched operations retrieve, update, or delete data in rows that meet some search criteria you specify. Through searched operations, the CA OPS/MVS product can process many rows of a table all at once.
- Cursor operations retrieve, update, or delete data in a range of rows that you select, one row at a time. To select rows, you point to them with a cursor.
- Table management operations manage your tables, including altering, creating, or deleting them.

More information:

Cursor Operations (see page 436)
Clauses Used in Searched Operations (see page 417)
Table Management Operations (see page 440)

What Are the Differences From Standard SQL?

The CA OPS/MVS version of SQL supports the tasks required to create, modify, access, and delete tables. It also has several features not offered by standard SQL:

- The CA OPS/MVS SQL converts numeric values to printable text strings, allowing you to compare numeric fields and character fields. After removing leading and trailing blanks from the character string and the converted numeric string, CA OPS/MVS compares the two strings.
- The CA OPS/MVS SQL always returns values passed to or fetched from SQL in a printable format. For example, CA OPS/MVS passes or retrieves values stored in binary format as a string of numeric digits and retrieves hexadecimal values in a printable X'*n*' format.

About the Sample Tables

The rest of this chapter describes what you can do with SQL statements and presents statement syntax examples. Many of these examples manipulate the sample relational tables named SYSTEMS and APPLICATIONS.

The Sample SYSTEMS Table

The SYSTEMS table contains this information:

ROW	NAME	CURRENT_ STATE	DESIRED_ STATE	RECOV_PROC
Row 1	CICS	DOWN	UP	FIXCICS
Row 2	IMS	UP	UP	FIXIMS

The Sample APPLICATIONS Table

The APPLICATIONS table contains this information:

ROW	APPL_ID	USER_ID	UPDATE	STATUS
Row 1	APPL1	TSOUSR1	1992-02-13	UP
Row 2	APPL2	TSOUSR2	1992-02-05	DOWN
Row 3	APPL5	TSOUSR8	1992-01-31	DOWN
Row 4	APPL10	TSOUSR22	1992-03-04	UP

Invoking SQL Statements

You can invoke SQL statements from any of these sources:

- A TSO terminal
- A TSO/E REXX program
- A TSO CLIST program
- An AOF rule running in either the production or the AOF test environment

  Note: You may use the AOF test environment to test rules that use ADDRESS SQL commands without having to set the Live Commands field on the AOF test panels to YES. All testing occurs on the private test copy of the Relational Data Framework database. For details, see the chapter “Using the OPSVIEW Editors Option” in the *OPSVIEW User Guide*.
- An OPS/REXX program

For examples, see the chapter “Relational Data Framework Reference” in the *Command and Function Reference*.

Restrictions for Cursor Operations

SQL statements that perform cursor operations may be invoked from all of the sources listed above *with the exception* of a TSO terminal.

More information:

How the Environment Determines Which Statements Are Permitted (see page 403)
Cursor Operations (see page 436)

Formats for Invoking SQL Statements

To invoke an SQL statement from a TSO terminal, a TSO/E REXX program, or a TSO CLIST, CA OPS/MVS provides the OPSQL command processor. The SQL statement has a maximum length of 2048 characters.

Use this format for OPSQL:

```
OPSQL sqlstatement
```

Note: Although the abbreviation SQL is an alias of the OPSQL command processor, we strongly recommend that you specify the complete text of the command. Doing so prevents confusion that can occur with other products having SQL command processors.

To invoke an SQL statement from an AOF rule (running under the production or the test facility) or an OPS/REXX program, CA OPS/MVS provides the ADDRESS SQL host environment. Use this format for ADDRESS SQL:

```
ADDRESS SQL
  "sqlstatement"
```

Diagrams that show the detailed syntax of each SQL statement (for example, ALTER TABLE, CREATE TABLE, and so on) appear later in this chapter. For a list of statements, see List of SQL Statements in this chapter.

How the Environment Determines Which Statements Are Permitted

The environment where you are using SQL determines the kinds of SQL statements you can invoke:

- Native TSO sessions or TSO running under ISPF cannot invoke SQL statements that perform cursor operations. Cursor operations deal directly with REXX and CLIST variables. Therefore, the CA OPS/MVS product does not allow you to invoke cursor operation statements from a TSO terminal.
- TSO/E REXX programs, TSO CLISTs, and OPS/REXX programs (including AOF rules) can invoke all of the SQL statements CA OPS/MVS supports, including statements that do cursor operations.

Destinations of SQL-related Error Messages

When an SQL statement is processed, it produces a return code that becomes the value of a variable named SQLCODE. When the value of the SQLCODE variable is not equal to 0, CA OPS/MVS writes error messages to a particular destination. The destination is dependent upon the environment from which the SQL statement was invoked.

When you use the ADDRESS SQL host environment to invoke an SQL statement from an AOF rule or OPS/REXX program, error messages are sent to the external data queue.

When you use the OPSQL command processor to invoke an SQL statement from:

- A TSO terminal, error messages are PUTLINEd to the terminal.
- A TSO/E REXX program, error messages are either trapped in stem variables or PUTLINEd to the terminal, depending on the current setting of the OUTTRAP() function.
- A TSO CLIST, error messages are either trapped in &SYSOUTLINE variables or PUTLINEd to the terminal, depending on the value of &SYSOUTTRAP.

Notes on Performing Cross-system SQL Operations

When you invoke an SQL statement using the ADDRESS SQL format, you can perform cross-system SQL operations.

Note the following:

- In terms of return codes and the SQLCODE, there are no differences between local and cross-system SQL operations. However, if you specify SYSTEM(EXT) or SYSTEM(ALL), the value of SQLCODE is always 0. You must specify each specific system, and then check the value of SQLCODE if you need to perform validity checking. For more information, see Return Codes from ADDRESS SQL Instructions in this chapter.
- When performing cross-system operations, error messages are sent to the external data queue.
- If you want, you can use the ADDRESS OPSCTL host command environment to send an SQL command to a remote system. This example illustrates the usage of ADDRESS OPSCTL in conjunction with ADDRESS SQL (where *sqlstatement* is any valid SQL statement):

  ```
  ADDRESS OPSCTL
    "MSF DEFAULT SYSTEM(SYS1) SYSWAIT(20)"
  ADDRESS SQL
    "sqlstatement"
  ```

  If you choose to use ADDRESS OPSCTL in this way, you may specify only one system name as the value for the SYSTEM keyword. For more information, see the *AOF Rules User Guide*.

List of SQL Statements

The SQL statement has a maximum length of 2048 characters. The following table describes the SQL statements the CA OPS/MVS product supports.

Note: For syntax information, see the chapter "Relational Data Framework Reference" in the *Command and Function Reference*.

ALTER TABLE

Adds columns to a table.

Type of operation: Table

Invoke from a TSO terminal: Yes

Invoke from a TSO/E REXX, TSO CLIST, or OPS/REXX program or from an AOF Rule: Yes

CLOSE

Ends a cursor operation

Type of operation: Cursor

Invoke from a TSO terminal: No

Invoke from a TSO/E REXX, TSO CLIST, or OPS/REXX program or from an AOF Rule: Yes

CREATE TABLE

Defines a new relational table to CA OPS/MVS.

Type of operation: Table

Invoke from a TSO terminal: Yes

Invoke from a TSO/E REXX, TSO CLIST, or OPS/REXX program or from an AOF Rule: Yes

DECLARE CURSOR

Defines a cursor operation.

Type of operation: Cursor

Invoke from a TSO terminal: No

Invoke from a TSO/E REXX, TSO CLIST, or OPS/REXX program or from an AOF Rule: Yes

DELETE FROM

Deletes the current row (in a cursor operation) or rows that meet your selection criteria (in a searched operation).

Type of operation: All

Invoke from a TSO terminal: Yes

Invoke from a TSO/E REXX, TSO CLIST, or OPS/REXX program or from an AOF Rule: Yes

DROP TABLE

Deletes a table.

Type of operation: Table

Invoke from a TSO terminal: Yes

Invoke from a TSO/E REXX, TSO CLIST, or OPS/REXX program or from an AOF Rule: Yes

FETCH

Retrieves all of the column values for the current row.

Type of operation: Cursor

Invoke from a TSO terminal: No

Invoke from a TSO/E REXX, TSO CLIST, or OPS/REXX program or from an AOF Rule: Yes

INSERT

Inserts a row into a table.

Type of operation: Table

Invoke from a TSO terminal: Yes

Invoke from a TSO/E REXX, TSO CLIST, or OPS/REXX program or from an AOF Rule: Yes

OPEN

Initiates a cursor operation and executes a SELECT statement.

Type of operation: Cursor

Invoke from a TSO terminal: No

Invoke from a TSO/E REXX, TSO CLIST, or OPS/REXX program or from an AOF Rule: Yes

SELECT

Retrieves column values in rows that meet your search criteria.

Type of operation: Searched

Invoke from a TSO terminal: Yes

Invoke from a TSO/E REXX, TSO CLIST, or OPS/REXX program or from an AOF Rule: Yes

UPDATE

Updates column values for the current row (in a cursor operation) or rows that meet your selection criteria (in a searched operation).

Type of operation: All

Invoke from a TSO terminal: Yes

Invoke from a TSO/E REXX, TSO CLIST, or OPS/REXX program or from an AOF Rule: Yes

Tools for Importing, Exporting, and Backing up Tables

The following section contains information on tools for importing, exporting, and backing up tables.

The READTBL and WRITETBL OPS/REXX Programs

You can use two OPS/REXX programs called READTBL and WRITETBL to do the following:

- Import and export Relational Data Framework tables from different systems.
- Make a backup copy of a table.
- Remove an unwanted column from a table.

Note: For more information, see the chapter “Relational Data Framework Reference” in the *Command and Function Reference*.

The OPCRTBDF Subroutine

The OPS/REXX subroutine, OPCRTBDF, is provided for user programs that may want to copy or save the definition of an existing table in the format of an SQL CREATE TABLE statement. OPCRTBDF analyzes the structure and defaults of an existing table and returns a return code and an SQL CREATE TABLE statement with the original table name or a new table name. The CREATE TABLE statement may be executed or saved as required. The keyword syntax for this program is contained in the comment section at the beginning of the program.

Storing Data in and Requesting Data From Relational Tables

You store automation data in a relational table in one of two ways:

- By specifying all of the data on an SQL UPDATE or INSERT statement.

  The following sample rule clause invokes an UPDATE statement that:

  - Looks for a row in the SYSTEMS table where the NAME column contains the value CICS
  - Assigns the values UP and DOWN to the CURRENT_STATE and DESIRED_STATE columns in that row

  ```
  ADDRESS SQL
    "UPDATE SYSTEMS SET CURRENT_STATE = 'UP',",
      "DESIRED_STATE = 'DOWN' WHERE NAME = 'CICS'"
  ```

  In this series of SQL statements, a table named TESTTBL is created, and two rows are inserted into it:

  ```
  ADDRESS SQL
   "CREATE TABLE TESTTBL (ACTION_NAME CHAR(17) NOT NULL PRIMARY KEY,",
     "ACTION_TYPE CHAR(8) NOT NULL PRIMARY KEY, ACTION_TEXT CHAR(200))"

  ADDRESS SQL
    "INSERT INTO TESTTBL (ACTION_NAME, ACTION_TYPE, ACTION_TEXT)",
      "VALUES ('UNKNOWN', 'TEST1', 'TSOCMD(OPSWTO TEXT(''UNKNOWN FIRED''))')"

  ADDRESS SQL
    "INSERT INTO TESTTBL (ACTION_NAME, ACTION_TYPE, ACTION_TEXT)",
      "VALUES ('DOWN_UP', 'TEST1', 'TSOCMD(OPSWTO TEXT(''DOWN_UP FIRED''))')"
  ```

- By having the SQL statement specify *host variables* containing the values you want to set. For example, the following sample command tells the CA OPS/MVS product to place REXX variable values in the CURRENT_STATE and DESIRED_STATE columns:

  ```
  ADDRESS SQL
    "UPDATE SYSTEMS SET CURRENT_STATE= "CICS.CUR",",
      "DESIRED_STATE = "CICS2.DES" WHERE NAME = '”CICS"'"
  ```

More information:

Description of Host Variables (see page 413)

How SQL Statements Pass Values to a Table

An SQL statement passes values in these ways:

- As strings. You can specify character, numeric, or hexadecimal strings:
  - Character strings-These strings always must be enclosed in single quotation marks. The CA OPS/MVS product does not translate the text in the quotation marks to uppercase characters, nor does it translate any variable names that appear in the text. For example, a character string might look like this:

    ```
    'This is a character string'
    ```

  - Numeric strings-A numeric string is a sequence of integers. To make the string represent a negative value (for example, -717), use a minus sign as a prefix.
  - Hexadecimal strings-You express a hexadecimal string as shown in the following example. The xxxx value represents hexadecimal digits.

    ```
    X'xxxx'
    ```

  Note: If a hexadecimal string has an odd number of hexadecimal digits, the digits are right adjusted to form the value. For example, X'123' is the same as X'0123'.
- As the value NULL. SQL defines this value as a null data value. SQL does not consider a zero-length character string to be a null value.
- As host variables. You can use dynamic variables or portions of variables in SQL statements you invoke from AOF rules or OPS/REXX programs. You also can use a host variable in any part of an SQL statement where you can use a string value.
- As a date, a time, a time stamp, or a packed decimal value. Specify these values as shown in the table that follows.

How SQL Processes Host Variables

If you defined a column as character data, SQL copies the data verbatim from the host variable. Therefore, the data SQL inserts into the table exactly matches the characters in the variable.

If you defined a column as numeric data, SQL assumes that the variable is a single sequence of digits that may have a leading minus sign. SQL ignores leading and trailing blanks.

If you defined a column as a hexadecimal value, SQL responds as follows:

- It accepts values in the form X'*xxxx*'.
- If the variable value is not in this form, SQL checks to see whether the value contains only hexadecimal digits 0-9, A-F, or both. If so, SQL converts the variable value into hexadecimal text.
- If the variable value does not have the form X'*xxxx*' or does not contain only hexadecimal digits, SQL sets the text exactly as is.

How SQL Processes Null Values

Standard SQL explicitly defines a null state for column values. This means that there may be no data value set for a column. A null column value for SQL is *not* the same as it would be for TSO/E REXX or a CLIST.

When a null table column is displayed, a four-byte NULL string is shown or represented in the variables.

When setting a column to a null value (that is, a zero length value), do *not* use a host variable. Rather, use the value NULL.

For example, to set a null column value in SQL, you could invoke the SQL UPDATE statement from a rule or an OPS/REXX program as follows:

```
ADDRESS SQL
  "UPDATE tablename SET colname = NULL..."
```

When SQL returns a null column value to CA OPS/MVS, it does so by returning a zero-length text string. This means that a null value looks just like an SQL column value that is of zero length. In SQL comparative expressions, though, a null column value is equivalent only to another null column value.

SELECT Statement—Request Data from a Table

You request data from a table through the SELECT statement. For example, to fetch the current state of the CICS system from the SYSTEMS table, you could include this line in an AOF rule:

```
ADDRESS SQL
  "SELECT CURRENT_STATE INTO :CUR_STATE FROM SYSTEMS",
    "WHERE NAME='CICS'"
```

The syntax for this statement breaks down like this:

- The ADDRESS SQL command invokes SQL to process the statement.
- The clause SELECT CURRENT_STATE INTO:CUR_STATE tells CA OPS/MVS to fetch an item from the CURRENT_STATE column and place it in a REXX variable called CUR_STATE.
- The clause FROM SYSTEMS WHERE NAME='CICS' tells SQL which item to fetch from which table. In this case, the item is the current state value from the row in the SYSTEMS table where CICS appears in the NAME column.

When this statement executes, CA OPS/MVS sets the following variables:

- CUR_STATE.0 = 1 (1 is the number of rows selected)
- CUR_STATE.1 = DOWN (DOWN is the value in the first row selected)
- SQLCODE=0 (0 is the result of the SELECT statement)

More information:

Specifying Stem Names (see page 414)

The ADDRESS SQL Environment and Host Variables

Issued through ADDRESS SQL statements, SQL commands such as SELECT and FETCH create stem REXX variables. For example, suppose that you have this statement:

```
ADDRESS SQL
  "SELECT NAME FROM MYTABLE"
```

The statement shown above generates these stem REXX variables:

- NAME.0 set to the value equaling the number of variables created
- NAME.1 through NAME.*n*, each of which is set to a value found in the NAME column of MYTABLE

Using the ADDRESS SQL statement above, you could write the following REXX code to display the retrieved NAME column information on your terminal, or to display error messages if the SELECT statement fails:

```
ADDRESS SQL
"SELECT NAME FROM MYTABLE"
IF SQLCODE ¬= 0 THEN
  DO
    SAY "ADDRESS SQL HOST COMMAND RET CODE =" SQLCODE
    DO WHILE QUEUED() <> 0
      PULL MESSAGES
      SAY  MESSAGES
    END
    RETURN
  END
SAY "Number of Variables Created =" NAME.0
DO I = 1 TO NAME.0
  SAY NAME.I
END
```

Description of Host Variables

When you fetch information from a relational table through a SELECT statement, CA OPS/MVS always returns values in the form of host variables unless you invoked the statement from a TSO terminal.

Note: If you invoke a SELECT statement manually, your TSO terminal displays the returned values.

What a host variable is depends on how you invoke an SQL statement. For instance, if you invoke a statement from a rule or an OPS/REXX program, then the host variables are OPS/REXX variables. Static variables can be used as host variables, but global and compound variables cannot.

However, you can use this technique:

```
ADDRESS SQL
  "SELECT XYZ FROM TABLEX WHERE COLDATA='"stem.abc"'"
```

In this example, REXX evaluates the compound symbol (stem.abc) because it is outside of the host command string, and substitutes its value in the SQL host command so that SQL sees a constant value rather than a host variable. This process is slightly more efficient than having SQL do the host variable resolution. When you use this technique, you may use global variables as well as compound symbols.

If you invoke a Statement from:

- TSO/E REXX
- Host variables are TSO/E REXX variables. OPS/REXX and TSO/E REXX resolve variables through different methods, so the two environments do not share common variables.
- A TSO CLIST

  Host variables are CLIST variables.
- A TSO Terminal Using the OPSQL TSO Command

  No variables exist in this context. Therefore, you cannot invoke SQL statements that require host variables (such as the SQL statements for cursor operations) outside of a CLIST or some type of REXX program.

A host variable can contain as many as 32,000 characters of data. If you provide no specific host variable names, CA OPS/MVS uses the names of the columns in the relational table as variable names. In an SQL statement, you indicate a host variable (where the host is a REXX program) by prefixing the variable name with a colon (:).

Specifying Stem Names

As illustrated by the example shown in Requesting Data From a Table in this chapter, for SQL statements that generate REXX variables, you can specify a stem name. For example, to fetch information about user names from the TABLE table and insert it into a series of variables with the stem USER_NAME, use an instruction like the following:

```
ADDRESS SQL
  "SELECT NAME INTO :USER_NAME FROM TABLE"
```

Executing this statement produces variable USER_NAME.0, which contains the number of variables produced by this instruction; and variables named USER_NAME.1 through USER_NAME.*n*, which contain user names taken from the database of the Relational Data Framework.

Return Codes from ADDRESS SQL Instructions

ADDRESS SQL instructions produce a return code in REXX variable RC. RC=0 indicates that the requested operation was successful. RC=4 indicates that the requested operation was at least partially successful, but that SQL issued a warning message. RC=8 indicates that the operation failed.

When RC=0, SQLCODE may be set to 0 or 100. An SQLCODE of 0 indicates that the operation was successful. An SQLCODE of 100 also indicates that the operation was successful and that the end-of-file (EOF) was reached as a result of the operation.

When RC=4 or 8, one or more messages are stacked in the External Data Queue (EDQ), and SQLCODE is set to a negative number that identifies the message number of the first message in the queue. For example, when the first queued message is OPS7407E, a second message, OPS7463E, may also be queued; however, SQLCODE is set to -7407.

When RC is greater than 8, it indicates that a problem occurred in the interface to SQL. The nature of the problem is indicated in the following table. EDQ and SQLCODE are not applicable when RC is greater than 8.

Return Code	Description	Program Action/Advice
0	Normal completion	Check SQLCODE for a possible EOF indication.
4	Warning completion	EDQ contains one or more messages describing the warning condition and SQLCODE indicates the number of the first message.

Return Code	Description	Program Action/Advice
8	An SQL error occurred	EDQ contains one or more messages describing the error condition, and SQLCODE indicates the number of the first message.
20	Interface error	The CA OPS/MVS subsystem is not active.
24	Interface error	A security rule or the security exit has rejected the SQL statement.
28	Interface error	At CA OPS/MVS startup time, a serious SQL database error was encountered. As a result, SQL is unavailable and all requests to it are denied.
40	Interface error	A serious control block error occurred.
50	Interface error	An internal SQL abend occurred.
54	Interface error	An abend occurred in either the security interface routine or the security user exit.
60	Interface error	The CA OPS/MVS product could not find a host variable specified in the SQL statement.
61	Interface error	The SQL statement specified an invalid host variable name.
62	Interface error	The SQL statement specified an invalid host variable name.
63	Interface error	The variable name is invalid because it did not conform to variable naming conventions.
64	Interface error	There is not enough storage to allow access to host variables.
80	Interface error	Invalid system parameter list.

Searched, Cursor, and Table Management Operations

The CA OPS/MVS product supports SQL statements that define tables to store automation data and add, change, or reference this data. These statements support three types of operations:

- Searched Operations (see page 416)

  Searched operations update or delete only the contents of rows that meet some search criteria you specify. SQL statements that perform searched operations can process many rows of a table at once.

- Cursor Operations (see page 436)

  Cursor operations update or delete the contents of a range of rows you select specifically by pointing to them with a cursor. For example, you could invoke a set of statements to update rows A through E in a table.

  Since cursor operations process one row at a time, they are useful for automation applications written in the REXX or CLIST languages where processing only one row at a time is desired.

- Table Management Operations (see page 440)

  Table management operations manage your tables, including altering, creating, or deleting them.

Searched Operations

Searched operations let you retrieve and subsequently update or delete data in rows that meet some search criteria you specify. You can use these search criteria to compare values in rows of one table, or to compare values in rows from different tables. You can invoke searched operations from any of these sources:

- TSO terminal
- TSO/E REXX program
- TSO CLIST
- AOF rule (running in either the production or the test environment)
- OPS/REXX program

Statements Used in Searched Operations

You can specify the WHERE clause and search criteria on four SQL statements-SELECT, UPDATE, DELETE FROM, and INSERT. When the data in a row meets the search criteria specified on a WHERE clause, SQL processes that data as follows:

- If a SELECT statement contains the WHERE clause, then SQL returns the data to the source that invoked the SELECT statement.
- If an UPDATE statement contains the WHERE clause, then SQL updates the rows containing data that matches the search criteria.
- If a DELETE FROM statement contains the WHERE clause, then SQL deletes the row containing data that matches the search criteria.
- If an INSERT statement contains the WHERE clause, then SQL retrieves the values to be inserted from rows matching the search criteria.

Clauses Used in Searched Operations

When performing search operations, you can use the following clauses:

- WHERE clause

  This is the most commonly used clause in searched operations. The WHERE clause allows you to specify the conditions that determine which data is to be retrieved and optionally updated or deleted. The WHERE clause is also used in cursor operations to search for data.

- ORDER BY clause

  This clause may be used in searched operations. The ORDER BY clause allows you to specify the order in which values are returned from tables.

Use the ORDER BY Clause to Arrange Values

When you select values from a table without specifying detailed search criteria (as you would with the WHERE clause), you can get a random listing of elements that are not in any particular order. If you have a large table, it can be difficult to locate the value you are after. The ORDER BY clause allows you to order the rows according to the values in one or more of the columns.

Review the following tips for using the ORDER BY clause:

- The data type of the column specified in the ORDER BY clause determines how the rows are ordered. For example, if the SELECT statement in the example had enacted the ORDER BY clause using the USER_ID column, the rows would have been arranged in alphabetical order, using the first characters found in the USER_ID column of each row.
- You can also arrange the values in descending order by using the DESC operator. This causes the values to be ordered with the largest value first, descending to the lowest value.
- The NULL value is higher than all other values for every data type.
- If the ORDER BY clause specifies a column that contains duplicate values, then rows that have the same ORDER BY value are arranged in random order.

To arrange values using the ORDER BY clause

1. The following statement selects all rows from the APPLICATIONS table:

   ```
   OPSQL SELECT * FROM APPLICATIONS
   ```

   This statement would retrieve all of the rows of the table. If the table happened to contain hundreds of entries, you might have a hard time locating a particular user.

2. Include the ORDER BY clause to order the listing by update time, starting with the oldest update:

   ```
   OPSQL SELECT * FROM APPLICATIONS ORDER BY UPDATE ASC
   ```

   Produces this result:

APPL_ID	USER_ID	UPDATE	STATUS
APPL5	TSOUSR8	2009-01-31	DOWN
APPL2	TSOUSR2	2009-02-05	DOWN
APPL1	TSOUSR1	2009-02-13	UP
APPL10	TSOUSR22	2009-03-04	UP

The rows are now arranged by ascending values in the UPDATE column. The ASC operator specified ascending order, which caused it to start with the lowest value and increase.

The data type of the column specified in the ORDER BY clause determines how the rows are ordered. For example, if the SELECT statement in the example had enacted the ORDER BY clause using the USER_ID column, the rows would have been arranged in alphabetical order, using the first characters found in the USER_ID column of each row.

You can also arrange the values in descending order by using the DESC operator. This causes the values to be ordered with the largest value first, descending to the lowest value.

The NULL value is higher than all other values for every data type.

If the ORDER BY clause specifies a column that contains duplicate values, then rows that have the same ORDER BY value are arranged in random order.

Use the WHERE Clause to Select Values

There are various tools that you can use to perform searches using the WHERE clause. Following is a list of each of these tools:

- A *predicate* is an expression that can be true or false and can contain either uppercase or lowercase characters. You can specify various types of predicates, including comparison predicates, the IN predicate, and the LIKE predicate.
- The *substring* function allows you to retrieve parts of character strings that are to be used as search criteria. The substring function may be used in conjunction with the predicates previously mentioned.
- A *join* allows you to select values from multiple tables to be used in your search. You need only specify the table names in your statement; the joining of the tables actually occurs internally in CA OPS/MVS.
- A *subquery* allows you to perform a query function that may be nested in another expression. Subqueries appear in parentheses in an SQL statement, and they can use all of the tools previously mentioned.
- *Aggregate functions* perform simple numeric calculations, mostly on values in a specified column in a table.

To select values in a WHERE clause

Use the following syntax:

```
WHERE
   [NOT]{predicate}
   [AND|OR [NOT] predicate]
```

Example: WHERE Clause

Suppose you want all rows from the APPLICATIONS table where the last updated date is equal to 2009-02-13 and the status is equal to UP. To select these rows, use this WHERE clause:

```
WHERE UPDATE = DATE '2009-02-13' AND STATUS = 'UP'
```

By including the NOT operator, you can use a WHERE clause to find all rows *except* those that were last updated on February 13, 2009:

```
WHERE NOT UPDATE = DATE '2009-02-13'
```

Use Comparison Predicates in WHERE Clauses

A comparison predicate is an expression that compares a column name to one of these values:

- Another column name
- A character string
- A numeric string
- A hexadecimal string
- A host variable name
- NULL

Operators for Comparison Predicates

A WHERE clause containing a comparison predicate uses this syntax:

```
WHERE colname relationaloperator value
```

The *relationaloperator* can be any of the following:

- = (equal)
- <> (not equal)
- > (greater than)
- >= (greater than or equal to)
- < (less than)
- <= (less than or equal to)

Example: Comparison Predicate

Suppose that you invoke this SELECT statement:

```
SELECT * FROM SYSTEMS WHERE CURRENT_STATE = 'UP'
```

In this statement, the search criteria is the predicate CURRENT_STATE = 'UP'. This predicate instructs SQL to select all rows in the SYSTEMS table where the predicate is true-that is, where the value in the CURRENT_STATE column equals UP. In the SYSTEMS table, the predicate is true only for the row containing information about IMS, so only that row is selected.

Now suppose that you alter the predicate in the SELECT statement above so that it becomes true when the value in the CURRENT_STATE column is **not** UP. Your revised statement might look like this:

```
SELECT * FROM SYSTEMS WHERE CURRENT_STATE <> 'UP'
```

The predicate in this new statement tells SQL to select all rows where the CURRENT_STATE column contains a value other than UP. In the SYSTEMS table, this predicate is true only for the row containing CICS information. Therefore, SQL selects only that row when it executes the revised SELECT statement.

Use IN Predicates in WHERE Clauses

The following section discusses the use of IN predicates in WHERE clauses, and using Boolean expressions.

Comparing One or More Values

An IN predicate compares the column name to one or more strings or host variable names. Use this syntax to specify an IN predicate:

```
WHERE colname IN (string | :hostvar,...)
```

Although you could compare a column name to multiple values using multiple Boolean ORs, do this more efficiently by including an IN predicate in your WHERE clause. For example, these two SELECT statements are equivalent:

```
"SELECT CURRENT_STATE INTO :SYSTEMS_UP",
  "FROM SYSTEMS WHERE NAME IN ('CICS','IMS','VTAM')"
```

```
"SELECT CURRENT_STATE INTO :SYSTEMS_UP",
  "FROM SYSTEMS WHERE NAME = 'CICS'",
    "OR NAME = 'IMS' OR NAME = 'VTAM'"
```

When either statement executes, SQL selects the values in the CURRENT_STATE column from all rows where the value in the NAME column is CICS or IMS.

Note: The SYSTEMS table does not contain a row where the NAME column has the value VTAM, so SQL ignores this part of the WHERE clause.

An IN predicate is true only if the column value exactly matches any of the specified strings or host variable values. For example, suppose that you invoke this SQL statement:

```
"SELECT CURRENT_STATE INTO :RECOV_PROC",
  "FROM SYSTEMS WHERE RECOV_PROC IN ('FIX')"
```

When this statement executes, it selects no rows from the SYSTEMS table because the text string (FIX) does not exactly match any values in the column named RECOV_PROC. However, if you rewrite the statement as follows, the predicate is true because the string matches the column contents. In this case, SQL selects both values in the RECOV_PROC column.

```
"SELECT CURRENT_STATE INTO :RECOV_PROC",
  "FROM SYSTEMS WHERE RECOV_PROC IN ('FIXCICS','FIXIMS')"
```

Using Boolean Expressions

If you want to, you can combine predicates into compound Boolean expressions. For example, you can create a WHERE clause like the following:

```
"WHERE (NAME = 'IMS' AND RECOV_PROC <> FIXCICS)",
  "OR DESIRED_STATE = 'DOWN'",
    "AND CURRENT_STATE IN ('UP','UNKNOWN')"
```

The CA OPS/MVS product supports up to three nested levels of Boolean expressions in parentheses.

Using LIKE Predicates in WHERE Clauses

The following section discusses the use of LIKE predicates in WHERE clauses, comparing character strings, and use of the ESCAPE keyword.

Comparing Character Strings

A LIKE predicate compares a column to a wildcard value you specify and selects columns conforming to the wildcard value. Use this syntax for a LIKE predicate:

```
WHERE colname LIKE 'wildcard'
```

You must enclose the *wildcard* value in single quotes. A wildcard can include any combination of characters and either the percent sign (%) or the underscore (_).

The percent sign denotes any set of characters, including blanks. For example, the following clause shown selects from the NAME column all of the following values: CIC, ACICX, ABCIC, CICXYZ.

```
WHERE NAME LIKE '%CIC%'
```

The underscore denotes only one character in a specific position. For each character of column data you do not want to match, the wildcard must include an underscore. For instance, to find the NAME column in the SYSTEMS table that contains exactly three characters with M as the middle character, use the following LIKE predicate:

```
WHERE NAME LIKE '_M_'
```

The predicate shown above finds the value IMS, the only value in SYSTEMS matching the wildcard. This predicate does not find other columns with M as their second data character, such as a column containing the text OMEGAMON. To find that row, you would use the following LIKE predicate:

```
WHERE NAME LIKE '_M_____'
```

Using the ESCAPE Keyword

When the percent sign or underscore character is part of the column data you want to match, add the ESCAPE keyword to your LIKE predicate. The ESCAPE keyword prevents SQL from interpreting the percent and underscore characters in column text as wildcards. For example, to find the column containing the characters I_MS, use the following predicate:

```
WHERE NAME LIKE 'I#_MS' ESCAPE '#'
```

Use Expressions and Functions

The following sections discuss how to use expressions, character-oriented functions, and numeric aggregate functions.

Expressions

An SQL expression is a series of operands related by arithmetic or character string operators that yields a value that can be used in other SQL clauses as an operand.

Valid operands for expressions are column names, host variables, literals, functions, and other expressions enclosed by parentheses. Valid operators for numeric expressions are (+, -, /, *) while character expressions have only the concatenation operator (||). If a value in an expression is NULL the expression value will be NULL. Numeric and character operands cannot be mixed in expressions. All operands must match the expected data type of the resulting value: numeric or character. Date, time, and timestamp data types are not supported in expressions. Like REXX the conventional rules of operator precedence and parenthetical evaluation order are applied.

The following are some examples of expressions:

Add 1 to an integer column value:

```
“UPDATE  TBL1 SET TOTAL_TIMES = TOTAL_TIMES + 1”
```

Use a function, literal, and character columns to produce a 'LAST NAME, FIRST NAME' value into a host variable:

```
“SELECT   TRIM(LAST_NAME) || ', ' || FIRST_NAME  INTO :NAME”
```

Introduction to Functions

Functions may appear in the column list of Select SQL statements or in any SQL statement clause where expressions are allowed such as the WHERE clause. Generally, the operands of a character string function can be literal character strings, host variables, or column names. Hexadecimal columns and literals are treated as character strings. When a function expression is used as a SELECT value in the column list, the resulting output variable stem name is *function name_n* or *function name_colname* depending upon the particular function and arguments used. The SQL INTO host variable list clause can be used to assign definite names to function result values. Compatible functions may also be nested. Any function value output variables will use the outermost function as the function name portion of the stem variable name.

Operands of functions should be of the data type expected by the function. If an SQL NULL column or value is used as an operand, the resulting value is always NULL. Except for the row COUNT function, aggregate functions such as AVG skip column values that are NULL. The function result is NULL when all the column values are NULL.

Character-oriented Functions

The optional SUBSTR keyword, specified as part of the search criteria on an SQL statement, allows that statement to examine or return parts of character strings. Use the following syntax for the SUBSTR keyword:

```
SUBSTR (colname FROM xx FOR yy)
```

The SUBSTR clause fetches from the named column yy characters starting with the character in position xx. For example, the following SUBSTR clause selects characters 5 through 8 from the RECOV_PROC column of the SYSTEMS table:

```
SUBSTR (RECOV_PROC FROM 5 FOR 3)
```

The TRIM Function

The TRIM function removes repeated occurrences of a specific character from the beginning and/or end of another character string. The return value of the TRIM function is the resulting character string. This function is very similar to the REXX STRIP function. The syntax for the TRIM function is:

```
TRIM(LEADING|TRAILING|BOTH 'trimchar' FROM 'target string'/colname)
```

The default value for the first keyword is BOTH and the default value for the trim character is a blank. The statement TRIM('*target string'/colname*) is equivalent to trimming leading and trailing blanks from a string value. The FROM keyword is only required when the trim character is specified.

In this example the character '2' is removed from the end of the USER_ID column value:

```
“SELECT APPL_ID  TRIM(TRAILING  '2' FROM USER_ID)
FROM APPLICATIONS WHERE NAME = 'APPL10'”
```

The REXX variable returned for the TRIM function is:

```
TRIM_USER_ID.1=TSOUSR
```

The POSITION Function

The POSITION function returns the integer value of the first location of a search character string within a target character string. If the search string does not occur in the target character string, zero is returned. A string of blanks is a valid search string. The syntax of the POSITION function is:

```
POSITION('search string'/colname  IN  'target string'/colname)
```

In the following example, the position of a literal string in a column is returned in a select statement:

```
“SELECT NAME POSITION('CICS' IN RECOV_PROC) FROM SYSTEMS”
```

This statement returns the following variables for the POSITION column:

- POSITION_RECOV_PROC.0=0
- POSITION_RECOV_PROC.1=3
- POSITION_RECOV_PROC.2=0

In the following example, the position of a column value in a literal value is used in the where clause of a select statement:

```
“SELECT  NAME  FROM SYSTEMS WHERE  POSITION( NAME IN 'XYZ IMS')
> 0”
```

Because the NAME column value of IMS does occur in the literal value specified, the NAME='IMS' row is selected.

The CHAR_LENGTH Function

The CHAR_LENGTH function returns the length of a character string literal or character column. For VARCHAR columns the actual used string length is returned. For CHAR columns the length of the column is returned. For literals the length of the literal is returned. If a column is NULL, NULL is returned. Output variable names for CHAR_LENGTH are shortened to simply LENGTH_n. The syntax for this function is:

```
CHAR_LENGTH('string'/colname)
```

In the following example, the TRIM function is combined with the CHAR_LENGTH function to find the used length of the column RECOV_PROC in each row. The following SQL statement returns the actual length of the data in the column for each row:

```
“SELECT CHAR_LENGTH(TRIM(RECOV_PROC)) FROM SYSTEMS”
```

The following array of variables is returned:

- LENGTH_1.0=2
- LENGTH_1.1=7
- LENGTH_1.2=6

The COALESCE Function

The COALESCE function returns the first non-null SQL value of a list of column names or literals. If all the operands are NULL, the result is NULL. Column and literal operands for this function must be of the same data type. The syntax is:

```
COALESCE('string'/colname, 'string'/colname, 'string'/colname, …)
```

In the following example, assume that only one of the columns of column names COL1, COL2, and COL3 contains a value for an application in any one row. The remaining columns are SQL NULL. All the columns could be NULL. The COALESCE function can display the one value in the columns or a default value.

```
“SELECT COALESCE(COL1, COL2, COL3, 'NOVALUE') FROM APPLTAB WHERE NAME='APPL10'”
```

Because this is the first COALESCE function reference in the SQL statement, the REXX variable stem created for the function value is COALESCE_1. If COL2=YES, then COALESCE_1.1=YES. If all the columns are NULL, COALESCE_1.1=NOVALUE.

The EXTRACT Function

The EXTRACT function returns the month, day, year, hour, minute, or second value from a date, time, or time stamp column value or literal. The syntax is:

```
EXTRACT(timeunit FROM date-time string/column)
```

where *timeunit* can have one of the following values:

- YEAR
- MONTH
- DAY
- HOUR
- MINUTE
- SECOND

In this example, rows whose last update month is February are selected using the extract function in the WHERE clause:

```
“SELECT NAME FROM APPLICATIONS WHERE EXTRACT(MONTH FROM UPDATE) = 2”
```

The APPL1 and APPL2 rows are returned.

The LOWER Function

The LOWER function translates a character string literal or column value to all lower case characters. This function's primary use is in SQL expression evaluation where character data in a column can be in mixed case. The syntax for this function is:

```
LOWER('string'/colname)
```

In the following example, the value of two columns assumed to be mixed-case are translated into lowercase for comparison.

```
“SELECT NAME FROM SYSTEMS WHERE LOWER(CURRENT_STATE) = LOWER(DESIRED_STATE)”
```

Only the NAME='IMS' row is selected.

The UPPER Function

The UPPER function translates a character string literal or column value to all upper case characters. This functions primary use is in SQL expression evaluation where character data in a column can be in mixed case. The syntax for this function is:

```
UPPER('string'/colname)
```

Example: UPPER function

In the following example the value of two columns assumed to be mixed-case are translated into uppercase for comparison.

```
“SELECT NAME FROM SYSTEMS WHERE UPPER(CURRENT_STATE) = UPPER(DESIRED_STATE)”
```

Only the NAME='IMS' row is selected.

Numeric Aggregate Functions

The following section provides a list of numeric aggregate functions, discusses return values based on numeric calculations, and provides usage information. For use as an example in the descriptions of the AVG, COUNT, MAX, MIN, and SUM functions, the following WORKSTATIONS table lists the workstations and their cost:

Workstation	Cost
Macintosh IIci	4840
NEC PC386	2520
IBM3270	1200
Macintosh IIcx	4840

Return Values Based on Numeric Calculations

There may be instances when you want to retrieve a value that is based on a mathematical calculation of a collection of values. Aggregate functions perform simple numeric calculations, mostly on values in a specified column in a table. SQL NULL column values are ignored for all aggregate functions except the row count function, COUNT(*).

For example, the following statement returns the average of the values in the COST column from the WORKSTATIONS table:

```
OPSQL SELECT AVG (COST) FROM WORKSTATIONS
```

Note: This particular function is to be used on numeric data type columns only.

When using the INTO keyword on a SELECT statement, the stem variables created use the specified variable stem name, as in the following example using the WORKSTATIONS table:

```
ADDRESS SQL
“SELECT COUNT (*) INTO VAR FROM WORKSTATIONS”
```

The above statement results in the following stem variables being created:

```
VAR.0=1 (the number of values returned)
VAR.1=4 (the number of rows in the table)
```

When the INTO keyword is not used on a SELECT statement, as in the following example:

```
ADDRESS SQL
“SELECT MIN (COST) FROM WORKSTATIONS”
```

The variables created are:

```
MIN_COST.0=1 (the number of values returned)
MIN_COST.1=1200 (the lowest value in the COST column)
```

Note that the compound variable stem names above are created by prefixing the aggregate function name (MIN), followed by an underscore (_), followed by the column name (COST).

List of Aggregate Functions

Keep this information in mind when using aggregate functions:

- You can use more than one aggregate function in a SELECT statement, but you must specify each function separately.
- To obtain the values of aggregate functions, your SELECT statement must include values for the INTO keyword. For example, consider the following:

  ```
  SELECT AVG (COST) COUNT (*) INTO :variable1 :variable2 FROM WORKSTATIONS
  ```

There are five aggregate functions provided with CA OPS/MVS:

- AVG-Returns the average value in a specified column
- COUNT-Returns the number of rows in a specified table
- MAX-Returns the maximum value in a specified column
- MIN-Returns the minimum value in a specified column
- SUM- Returns the sum of all values in a specified column

AVG Function

The AVG function returns the average of all values in a specified column. For example, to obtain the average cost of all workstations in the table named WORKSTATIONS, you could use the following statement:

```
OPSQL SELECT AVG (COST) FROM WORKSTATIONS
```

This function would yield a result of 3350.

The AVG function is limited to use on numeric data types only.

COUNT Function

The COUNT function returns the number of rows in a specified table. In the simplest example, you could count all of the rows in the APPLICATIONS table with this statement:

```
OPSQL SELECT COUNT (*) FROM WORKSTATIONS
```

You can also use the WHERE clause in conjunction with COUNT to narrow the scope of your tally. For example, to count the number of applications in the APPLICATIONS table that user TSOUSR1 altered, you would use this statement:

```
OPSQL SELECT COUNT (*) FROM WORKSTATIONS WHERE COST < 3000
```

MAX Function

The MAX function returns the maximum value in a specified column.

To find the maximum cost of a workstation in the WORKSTATIONS table, you could use this statement:

```
OPSQL SELECT MAX (COST) FROM WORKSTATIONS
```

This yields an answer of 4840.

As you may have noticed, there are two workstations with a cost of 4840. Suppose you want to retrieve the station and the cost of all of the workstations that have that maximum cost. You could accomplish this by nesting a subquery in your statement as follows:

```
ADDRESS SQL
 "SELECT STATION, COST FROM WORKSTATIONS",
   "WHERE COST = (SELECT MAX (COST) FROM WORKSTATIONS)"
```

This yields the following result:

Workstation	Cost
Macintosh IIci	4840
Macintosh IIcx	4840

MAX is not limited to numeric values, contrary to what it would seem. You can also use it on characters and dates. When used with character data, MAX means last in alphabetical order.

MIN Function

The MIN function returns the minimum value in a specified column. For example, to find the minimum cost of a workstation in the WORKSTATIONS table, use this statement:

```
OPSQL SELECT MIN (COST) FROM WORKSTATIONS
```

This yields a result of 1200.

You can also combine the MAX and MIN functions in a single statement, as follows:

```
ADDRESS SQL
"SELECT MAX (COST), MIN (COST) FROM WORKSTATIONS"
```

This yields a result of 4840 and 1200.

As with MAX, the MIN function is not limited to numeric data. When used with character data, MIN means first in alphabetical order.

SUM Function

The SUM function returns the sum of all values in a specified column. For example, to obtain the total cost of all workstations in the WORKSTATIONS table, use this statement:

```
OPSQL SELECT SUM (COST) FROM WORKSTATIONS
```

This would yield a result of 13,400.

The SUM function is limited to use on numeric data types only. All integer and decimal data types are accepted.

Join Operations

SELECT statements can search for data stored in more than one relational table. Selecting data from more than one table is called a *join operation*; it produces a new temporary table containing all rows and columns of all tables referenced in the SELECT statement. This temporary table remains in memory only until the SELECT statement finishes executing.

The following section discusses comparing values from multiple tables, and defining aliases or correlation values for table names.

Compare Values from Multiple Tables

A single SELECT statement can now contain as many as eight different table references. Each usage of the same table name in a single statement counts as a separate table reference.

When using more than one table name in a SELECT statement, consider whether any column references in that statement clearly indicate to which table each column belongs. Where necessary, indicate which table each referenced column belongs to by specifying column names as follows: *tablename.colname*. A period must separate the table and column names.

Define Aliases or Correlation Values for Table Names

Referencing several tables in a SELECT statement can make that statement long and complex. So, you can define aliases or correlation values for table names, using the AS keyword as shown in the following example:

```
FROM tablename1 AS alias1, tablename2 AS alias2
```

You may use a single blank in place of the AS keyword to define an alias or correlation value for a table. For example, the following statement is equivalent to the preceding example:

```
FROM tablename1 alias1, tablename2 alias2
```

A comma separates distinct table names, in contrast to a blank without a comma, which separates the original table name from its alias or correlation value.

Once you define an alias or a correlation value for a table name, you can use it in subsequent parts of your SELECT statement. For example, a WHERE clause like the one below might follow the clause defining alias1 and *alias2*:

```
WHERE alias2.colname1 = tablename2.colname2
  AND alias1.colname1 = tablename3.colname3
```

Using Subqueries

The following section discusses using subqueries. It explains the purpose of using subqueries to reduce the amount of table data returned, and provides an example of a subquery being used on the SELECT statement.

Reduce Amounts of Data Returned

On any SQL statement that specifies search criteria, you can use subqueries to reduce the amount of table data returned. A subquery is a SELECT statement that is part of the WHERE clause of another statement.

A subquery allows CA OPS/MVS to make decisions based on data values from tables other than the table currently being accessed. For example, you might want to extract an application name from the current table only if that name matches a value from a column in another table.

The following SELECT statement uses a subquery to compare data from the CURRENT_STATE and DESIRED_STATE columns of the SYSTEMS table to the contents of the CURRENT_STATE column of another table called CICS_TABLE:

```
ADDRESS SQL
  "SELECT CURRENT_STATE, DESIRED_STATE FROM SYSTEMS",
    "WHERE CURRENT_STATE = (SELECT CURRENT_STATE FROM",
      "CICS_TABLE WHERE NAME = 'CICSA')"
```

In the above statement, the subquery defines the current state value for resource CICSA as the value which data fetched from the SYSTEMS table must match. Assuming that CICSA has a current state of UP, this statement fetches from SYSTEMS the current and desired state values from row 2 (both also UP).

A subquery must fetch only one piece of table data, except when the WHERE clause using that subquery also contains an IN keyword. Subqueries can operate on more than one column or row. For instance, the following INSERT statement copies every row from CICS_TABLE into the SYSTEMS table, so long as the column definitions for both tables are compatible and defined in the same order:

```
ADDRESS SQL
  "INSERT INTO SYSTEMS SELECT * FROM CICS_TABLE"
```

Cursor Operations

SQL statements typically alter or fetch data from many rows of a table at once. If 2,000 rows satisfy the search criteria you specify on a statement, CA OPS/MVS processes all 2,000 rows. However, this complicates matters when you invoke SQL statements from OPS/REXX. When you do not know how much output a statement will generate, it may become inefficient to assign that output to REXX variables.

SQL (and the CA OPS/MVS product) resolve this problem through a set of statements that manage cursor operations. In a cursor operation, you specify a range of rows to update or delete. Starting with the first row, CA OPS/MVS then updates or deletes each row one at a time until it processes all rows in the range.

You can execute cursor operations from a TSO/E REXX program, a TSO CLIST, an OPS/REXX program, or an AOF rule. You cannot execute cursor operations from a TSO terminal.

Statements Used in Cursor Operations

A cursor operation consists of these SQL statements, which you must invoke in the following order:

1. The DECLARE CURSOR statement. Before you start a cursor operation, you must invoke a DECLARE CURSOR statement. This special form of the SELECT statement defines a **cursor** to point to the rows you want to process. This cursor is a key that you pass to CA OPS/MVS in subsequent statements so that those statements act upon the proper rows. The CA OPS/MVS product holds this statement in reserve until you are ready to process the selected rows.
2. The OPEN statement. To start the cursor operation, you invoke the OPEN statement. This statement tells CA OPS/MVS to select the first of the rows specified on the DECLARE CURSOR statement.
3. The FETCH statement. Once you have opened the cursor, the FETCH statement fetches the values in the first selected row. Using the FETCH statement to access a row makes that row current. You can then invoke the UPDATE or DELETE FROM statement to modify or delete that row.

   CA OPS/MVS executes one FETCH statement for each row to be fetched, until it has processed all of the rows in your table or you end the cursor operation.
4. The UPDATE or DELETE FROM statement. The UPDATE and DELETE FROM statements work in cursor operations as they do when you invoke them alone, with one exception. When an UPDATE or DELETE FROM statement invoked as part of a cursor operation contains a WHERE clause, that clause instructs the CA OPS/MVS product to update or delete the currently selected row.

5. The CLOSE statement. This statement ends the cursor operation, telling CA OPS/MVS to stop processing rows. If you omit the CLOSE statement, the update or delete operation continues until CA OPS/MVS finishes processing all of the rows in the table. When CA OPS/MVS processes the last row, it sets the SQLCODE global variable to 100.

Example: Syntax for Cursor Operations

Because the statements that control cursor operations function in sets, the examples that follow show groups of statements you invoke together to update or delete rows one at a time.

Note: You may specify the text ADDRESS SQL on each statement, as shown in Example 2. However, since these statements should follow one another in the order shown, it is *necessary* to specify the text ADDRESS SQL *only* on the DECLARE CURSOR statement, as shown in the following example.

- To perform a cursor update operation, invoke this set of statements from an OPS/REXX program:

```
ADDRESS SQL
  "DECLARE cursorname CURSOR FOR selectstatement"
  "OPEN cursorname"
  "FETCH cursorname INTO hostvarlist"
  "UPDATE tablename
    SET colname = string | :hostvar | NULL
    [, colname = string | :hostvar  | NULL]
    WHERE CURRENT OF cursorname"
  "CLOSE cursorname"
```

  For descriptions of the operands, see the text following Example 2.

- To perform a cursor delete operation, invoke this set of statements from an OPS/REXX program:

```
ADDRESS SQL
  "DECLARE cursorname CURSOR FOR selectstatement"
ADDRESS SQL
  "OPEN cursorname"
ADDRESS SQL
  "FETCH cursorname INTO hostvarlist"
ADDRESS SQL
  "DELETE FROM tablename WHERE CURRENT OF cursorname"
ADDRESS SQL
  "CLOSE cursorname"
```

cursorname

This is the 1- to 18-character name of the cursor (pointer) that provides a common reference point for a set of related SQL cursor operation statements.

selectstatement

This is the text of a SELECT statement to execute when you invoke the OPEN statement. This SELECT statement should use the syntax described in SELECT Statement in this chapter.

hostvarlist

Used on the FETCH statement, the *hostvarlist* variable identifies a set of host variable names to store the selected column values. The order in which you specify REXX stem names should correspond to the order in which column names are specified on the DECLARE CURSOR statement.

tablename

This is the table containing the rows to be updated by the UPDATE statement or deleted with the DELETE FROM statement.

Guidelines for Writing Cursor Operation Statements

The following REXX program outline demonstrates how you might use SQL cursor operations.

Suppose that you want to examine the SYSTEMS table and write a message to the operator each time you find a system whose current state is DOWN but should be UP. You can do this using an SQL cursor operation, invoked by SQL statements in an OPS/REXX program as follows:

```
(first few lines of program)
.
.
ADDRESS SQL
  "DECLARE X CURSOR FOR"
  .
  .
ADDRESS TSO
  (TSO command)
  .
  .
ADDRESS SQL
  "OPEN X"
  .
  .
  "FETCH X INTO NAME"
```

Once the DECLARE CURSOR statement invokes the SQL host environment, subsequent SQL statements do not need to include the text ADDRESS SQL as long as the program does not switch to another default host environment.

However, because a statement invoking another host environment appears in the program above (the ADDRESS TSO statement shown before the SQL OPEN and FETCH statements), you have to invoke the TSO host environment with an ADDRESS statement and then switch back to the SQL host environment.

You can explicitly declare the host environment on each SQL statement by using the following form:

```
ADDRESS SQL 'sqlstatement'
```

It is a good idea to use the above form in a large program or a program that sends commands to several different host environments, since you will never send the wrong command to the wrong environment.

OPS/REXX Program That Demonstrates Cursor Operations

This sample program demonstrates the use of cursor operations:

```
/*-------------------------------------------------------*/
/* Declare a cursor that will select 3 columns from any */
/* "broken" (CURRENT <> DESIRED) DB2 regions.           */
/*-------------------------------------------------------*/
ADDRESS SQL 'DECLARE CSR1 CURSOR FOR',
  'SELECT NAME CURRENT_STATE DESIRED_STATE FROM STCTAB',
  "WHERE CURRENT_STATE <> DESIRED_STATE AND TYPE = 'DB2'"
IF sqlcode <> 0 THEN CALL SQLERROR /* Check return code!*/
ADDRESS SQL 'OPEN CSR1'    /* SQL will now get the data */
IF sqlcode = 100 THEN      /* Were any rows selected?   */
  DO                       /* No, issue msg & exit      */
    SAY 'All resources in table are at desired state'
    EXIT 0
  END
IF sqlcode <> 0 THEN CALL SQLERROR /* Check return code!*/
/*-------------------------------------------------------*/
/* Fetch each selected row, one row at a time.          */
/*-------------------------------------------------------*/
DO WHILE SQLCODE <> 100
  ADDRESS SQL 'FETCH CSR1 INTO :NAME, :CURRENT, :DESIRED'
  IF sqlcode = 100 THEN LEAVE      /* No more rows, done*/
  IF sqlcode <> 0 THEN CALL SQLERROR /* Check ret code! */
  /* Note that name.1, is returned, not just "name".    */
  SAY 'NAME='name.1',',
      'CURRENT='current.1', DESIRED='desired.1
```

```
  /*-----------------------------------------------------*/
  /* If CURRENT_STATE is neither UP nor DOWN, update     */
  /* this row in the table to set it to UNKNOWN.         */
  /*-----------------------------------------------------*/
  IF WORDPOS(current.1,'UP DOWN') = 0 THEN
    DO
      /* Set host variable.  This must be a simple      */
      /* variable (CURRENT), not a stem (like CURRENT.1)*/
      current = 'UNKNOWN'
      ADDRESS SQL 'UPDATE STCTAB',     /* Perform update*/
        'SET CURRENT_STATE = :current',
        'WHERE CURRENT OF CSR1'
      IF sqlcode <> 0 THEN
        CALL SQLERROR                  /* Check ret code*/
    END
END
ADDRESS SQL 'CLOSE CSR1' /* Be sure to close cursor     */
EXIT 0
/* Subroutine to display diagnostic data and exit       */
SQLERROR:
PARSE SOURCE . . pgm .
SAY 'SQL error in program 'pgm' called from line 'sigl
SAY 'RC='rc', SQLCODE='sqlcode
ADDRESS SQL 'CLOSE CSR1' /* Be sure to close cursor!    */
EXIT 12
```

Table Management Operations

The following section discusses table management operations.

Where to Perform Table Management Operations

Using the OPSQL command processor, you can perform table operations from a TSO terminal, a TSO/E REXX program, or a TSO CLIST program. Using the ADDRESS SQL host environment, you can perform table operations from an AOF rule or any other OPS/REXX program.

Table Management Statements

The following statements let you perform table management operations:

ALTER TABLE

Adds columns to a table

CREATE TABLE

Defines a new table

DELETE FROM

Deletes rows from a table

DROP TABLE

Deletes a table

INSERT

Inserts rows in a table

UPDATE

Updates column values in a table

For more information about the statements described here, see the *Command and Function* Reference.

Add Table Columns

To add a column to a table, use the ALTER TABLE statement.

Suppose the APPLICATIONS table needs a column added to it called RET_CODE, which holds a return code for the last execution of the application. To add this column, issue this command:

```
OPSQL ALTER TABLE APPLICATIONS ADD COLUMN RET_CODE CHAR(2)
```

This example uses a CHAR (character) data type definition, specifying that the column being added is 2 characters in width. Other valid data types for columns include integer, small integer, decimal, hexadecimal, date, time, and time stamp.

Define a New Table to the Product

To define a new table, use the CREATE TABLE statement.

For example, this statement defines the SYSTEMS table.

```
ADDRESS SQL
  "CREATE TABLE SYSTEMS (NAME CHAR(8) PRIMARY KEY,",
                        "CURRENT_STATE CHAR(4),",
                        "DESIRED_STATE CHAR(4),",
                        "RECOV_PROC    CHAR(8))"
```

Delete Table Rows

To delete rows from a table, use the DELETE FROM statement.

For example, to delete all rows from the table APPLICATIONS when the status is DOWN, issue this command:

```
ADDRESS SQL
  "DELETE FROM APPLICATIONS WHERE STATUS = 'DOWN'"
```

Delete a Table

To delete a relational table, use the DROP TABLE statement.

For example, to delete the APPLICATIONS table, issue this command:

```
OPSQL DROP TABLE APPLICATIONS
```

Insert a Row into a Table

To insert a row into a relational table, use the INSERT statement.

For example, to add a row into the APPLICATIONS table, issue this command:

```
ADDRESS SQL
  "INSERT INTO APPLICATIONS",
    "(APPL_ID, USER_ID, UPDATE, STATUS)",
      "VALUES ('APPL33' 'TSOUSR33' DATE '1992-04-15' 'DOWN')"
```

Update Values in a Table

To update values in a table, use the UPDATE statement.

Suppose you want to update values in the table APPLICATIONS when the STATUS column has a value of DOWN. To change the status to UP for these rows, issue this command:

```
OPSQL UPDATE APPLICATIONS SET STATUS = 'UP' WHERE STATUS = 'DOWN'
```

Use the Relational Table Editor Batch API

You can use the relational table editor OPS/REXX program, ASOTEAPI, in a batch API mode to perform some of the operations that are currently part of the ISPF menu-driven interface of OPSVIEW option 2.6. Using this API interface instead of conventional ADDRESS SQL statements can reduce your amount of effort in copying or changing table structures.

For a description of the batch API functions provided by the ASOTEAPI OPS/REXX program, see the chapter “Relational Data Framework Reference” in the *Command and Function* Reference.

Maintaining Cross-system Serialization

To prevent the simultaneous updating of a relational table by more than one user, the batch API mode of ASOTEAPI uses the same serialization mechanism as the OPSVIEW relational table editor. This pseudo ENQ/DEQ mechanism is implemented using global variables, maintaining cross-system serialization. Thus, any attempt to change a table using the batch API fails if the table is in use by the ISPF relational table editor user or another batch API operation. Therefore, you should generally limit the primary use of the batch API to occasional table modification operations that do not conflict with normal table usage. If you choose to use the batch API as a part of your production operations, you should anticipate possible serialization conflicts in the automation procedure code.

Duplicate Keys

When the batch API is executed in an active ISPF environment, it uses ISPF message services. When the transfer of relational table rows results in duplicate keys, a panel may be displayed in the TRANSFER API. Without an active ISPF environment, all messages are issued using a REXX SAY statement and the duplicate key transfer operation fails. Therefore, you should be aware of the ISPF status of the environment in which the batch API is running.

Chapter 12: Editing Relational Tables

This section contains the following topics:

Use the Relational Table Editor

CA OPS/MVS provides a relational table editor to simplify saving, updating, and fetching data from relational tables. These relational tables organize the information about your system that rules and automation procedures collect. You create relational tables through the CA OPS/MVS Relational Data Framework facility, described in the chapter "Using the Relational Data Framework."

The table editor is an ISPF application you can access from the primary menu of the CA OPS/MVS OPSVIEW Interface. During a table editing session, you can use ISPF dialogs and panels and ISPF-like edit commands to:

- Display a list of existing relational tables.
- Copy an existing table and rename it.
- Create a new table.
- Edit the contents of a table.
- Delete a table.
- Rename a table.

Note: When you add rows to or delete rows from a table and save your changes, CA OPS/MVS rearranges the rows in one of these ways:

- If the table has a primary key, CA OPS/MVS arranges the rows in alphabetical order by key.
- If the table has no primary key, CA OPS/MVS arranges the rows in the order in which they were added.

  Note: CA OPS/MVS appends the most recently added rows to the end of the table.

To access the table editor:

1. Choose CA OPS/MVS from your primary ISPF menu. The CA OPS/MVS OPSVIEW Primary Option Menu panel appears.
2. Select option 2, Editors.
3. Select option 6, for Table Edit.

Important! The table editor prevents multiple users from editing the same relational table simultaneously. However, the SQL rules keyword and the SQL, STATESET, and OPSSMTBL command processors do not prevent simultaneous editing of a table. Therefore, during your table editing session, rules, CLISTs, REXX EXECs, or other users may be making changes to the table you are editing but you cannot see these changes.

When you access the table editor, CA OPS/MVS displays the RDF Table Editor Primary Panel, shown here:

```
RDF Table Editor --------------- Primary Panel --------------------------------
OPTION ===>
       B - Browse this table        C - Copy table
       E - Edit table               R - Rename table
       I - Insert new table         F - Free table
       D - Delete table             T - Transfer table contents

                    blank - Display table list

SPECIFY RELATIONAL TABLE (see note below):
   NAME     ===> _____________________________  (Required for B, C,D,E,F,R,T)
   NEWNAME  ===> _____________________________  (Required for C,I,R,T)
CONFIRM DELETES: YES (Enter YES to require delete confirmation)
NOTE: To use a table on another system specify the table name as system>table
      Specify ? as the system name to get a list of all systems.
```

This is the main menu for the table editor.

By default, CA OPS/MVS asks you to confirm any requests to delete information from a table. To delete information without confirming delete requests, specify NO in the Confirm Deletes field, as shown on the panel above, instead of YES.

To edit a table, do either of the following:

- Type one of the edit option commands described on this panel.
  CA OPS/MVS prompts you to enter the name of the table to copy, delete, edit, free, browse, or insert in the Name field. The table name goes into the Newname field when you copy a table. After typing the table name, press Enter.

- Press Enter without entering a command. CA OPS/MVS displays its Table List panel. From there, you can choose a table to edit; following is an example of this panel:

```
RDF Table Editor ------ RDF Table List for System OPSQA ---------  ROW 1 OF 22
COMMAND ===>                                                  SCROLL ===> PAGE
    OPTIONS:  Browse Copy Delete Edit Free Insert Rename S(edit) Transfer
 Sel Table                       New Table
 ___ COLUMN                      ____________________________
 ___ ABCXYGTBL                   ____________________________
 ___ DCXA                        ____________________________
 ___ DCYZSTC                     ____________________________
```

This panel lists all relational tables in use at your site and the system where each table resides.

When working with the Relational Table List panel, you can issue these primary commands from the command line:

DOWN

Scrolls toward the bottom of the table list.

UP

Scrolls toward the top of the table list.

END

Cancels all pending line commands and exits from this panel.

Use Edit Option Commands

Both the RDF Table Editor Primary Panel and the Table List panel use the edit option commands listed just below the command line. If you use the RDF Table Editor Primary Panel, type these commands from the command line and press Enter. If you use the Table List panel, type one of these commands to the left of the table you want to edit and press Enter:

B

Browse this table. None of the columns are modifiable. When you enter the B command, CA OPS/MVS displays the Table Data Editor panel shown in Editing the Contents of an Existing Table in this chapter. The RDF TABLE and COLUMN tables are always displayed in browse mode.

C

Copies this table. If you enter this command from either panel, CA OPS/MVS requires you to type the name of the new table in the Newname field.

Specifying the system name with the table name is optional. If you specify an asterisk in place of a system name, the table editor displays its System List panel and allows you to choose one system from the list.

D

Deletes this table. CA OPS/MVS asks you to press Enter a second time to confirm that you want to delete the table.

E

Edit (modify) this table. When you enter the E command, CA OPS/MVS displays the Table Data Editor panel shown in Editing the Contents of an Existing Table in this chapter. From this panel, you can edit the table contents.

Note: To edit a new table, issue the I command instead of the E command.

F

Frees a table. This command releases a table enqueued by another user. This other user may be on a remote system.

In response to the F command, CA OPS/MVS displays a panel (not shown) reporting which user on which system has locked the table and asking you to notify that user before confirming your request to free the table. If the remote user tries to save the table after you have freed it, the table editor rejects the save request.

I

Insert a table. This command causes CA OPS/MVS to display a panel that asks you to do one of the following:

Tell CA OPS/MVS to give your new table the standard column definitions for a System State Manager table.

Enter the name of an existing table that contains the column definitions you want to use for your new table. Specifying the system name with the table name is optional. If you specify an asterisk in place of a system name, the table editor displays its System List panel and allows you to choose one system from the list.

After you tell CA OPS/MVS how to define columns for your new table, CA OPS/MVS displays the Edit Table Structure panel.

Important!

- When inserting a new table, do *not* select a table name beginning with the characters ATM. This is a prefix, and causes the table to be hidden when you view tablenames from OPSVIEW.
- When defining a new table, be sure to define all of the rows and columns you need. If you forget to add one or more rows or columns when you create a table, or you want to change the structure of a table, or both, you can do so only through one of these methods:
  - From a CLIST, a REXX EXEC, or foreground TSO, invoke the ALTER TABLE SQL statement.
  - From the table editor, issue the D command to delete the table in error. Then, issue the I command and redefine the table.

R

Rename a table. CA OPS/MVS asks you for a new table name. Specifying the system name with it is optional. If you specify an asterisk in place of a system name, the table editor displays its System List panel and allows you to choose one system from the list.

blank

If you press Enter from the RDF Table Editor Primary Panel without specifying a table name, the table editor displays a list of available tables.

S

Select a table. This command takes the same actions as the E (Edit) command.

Point-and-shoot is enabled to issue the E or S line command for any displayed table. To issue the E or S line command for a displayed table using the point-and-shoot method, place the cursor in the Sel column to the left of the table name and press the ENTER key. Point-and-shoot is enabled only if no primary or line commands have been entered.

Protecting System State Manager Tables

The active System State Manager directory table specified by the STATETBL parameter may not be modified using any table editor commands in ISPF on-line mode. Use the OPSSMTBL TSO command and OPS/REXX function or the System State Manager control panel (4.11.1 of OPSVIEW) to modify this table. Browse is the only table editor command that you can use on System State Manager directory tables.

Active System State Manager resource tables are protected from potentially destructive command operations of the RDF table editor by requiring or requesting that the target System State Manager resource table be removed from System State Manager control before performing an operation that could damage the logical integrity of the resource table. Potentially destructive operations include:

- Delete a System State Manager resource table
- Rename a System State Manager resource table
- Insert a System State Manager resource table

  Note: Insert implies that this table does not currently exist.
- Copy a table into a System State Manager resource table
- Transfer a table into a System State Manager resource table

When such an operation is detected, a panel is displayed, asking you if you want to remove the System State Manager resource table from System State Manager control. Press Enter to accept the default (Y/YES). The table is removed from System State Manager control, and the table editor operation is completed.

Note: You can optionally bypass the removal of a resource table from System State Manager for the *copy* and *transfer* operations by the entering N (NO) on this panel.

If the System State Manager resource table was removed from System State Manager control, a panel is displayed, asking you if you want to return the modified resource table to System State Manager control. Press Enter to accept. The resource table is returned to System State Manager with the same properties it had when it was removed.

Note: When a resource table is returned to System State Manager control using the OPSSMTBL command, both the current and desired states are set to the UNKNOWN state. The desired state must be reset to the correct value using either the checkpoint state value, the System State Manager schedule manager reset function, or by some programmatic or manual procedure.

When you start the table editor OPS/REXX program ASOTEAPI in batch API mode, protection of System State Manager resources tables is controlled by the System State Manager PROT keyword of the API command.

Note: For details on this keyword, see the *Command and Function Reference*.

Edit the Structure of a New Table

You can use the following Table Structure Editor panel to create a new relational table:

```
Table Structure Editor ---------- SSM>XYZ----------------- COLUMNS 00001 00072
Command ===>                                                 Scroll ===> PAGE
COL--> COLUMN-NAME        DATA-TYPE      ATTRIBUTES  DEFAULT
****** ***************************** TOP OF DATA ******************************
000001 TABLE_ID           CHAR(4)        PK
000002 TABLE_NAME         CHAR(18)
000003 DESIRED_STATE      CHAR(8)        NOT NULL    UP
000005 DATA_TYPE          HEX(2)
000006 DATA_OFFSET        SMALLINT
000007 DATA_LENGTH        SMALLINT
000008 SCALE              SMALLINT
000009 AVMAN              HEX(4)
****** *************************** BOTTOM OF DATA ***************************
```

From this panel, you configure columns for a new table. This sample display shows the contents of a table called SSM>XYZ.

When defining a new table through the Table Structure Editor panel, follow the same requirements that you would follow if creating the table using the CREATE TABLE statement of SQL.

Note: For a detailed discussion of these requirements, see the *Command and Function Reference*.

Special Criteria for Column Descriptions

For a column to be null, the first byte of that column must contain the word NULL. However, you receive error messages if NULL appears in the first column byte and you defined that column as NOT NULL. A column defined as NOT NULL can contain the word NULL only if the word begins after the first byte.

If you edit data in the DEFAULT column of a table, CA OPS/MVS reads that data exactly as you enter it. For example, if you are using the Table Structure Editor panel and you insert two blanks and a word into the DEFAULT column, the first two characters in the DEFAULT column remain blank.

Primary Commands for Creating Table Structure

To manage your table creation session, you can issue the following primary commands (similar to ISPF edit commands) from the command line. If a command appears in all uppercase letters, you must enter the full command name. Otherwise, enter only the part of the command shown in uppercase.

CANcel

Exits this table creation session without creating the table.

CAPS

Converts the column names you enter to uppercase or lowercase characters. If you enter the column name in uppercase, the table editor converts data entered in that column to uppercase. If you enter the column name in lowercase, data entered in that column remains in the case in which you entered it.

The CAPS command applies only to columns containing character data.

Note: By default, CA OPS/MVS sets the names of columns with non-character data to uppercase.

Issue one of these versions of the CAPS command:

- CAPS ON-Sets the names of all character columns to uppercase
- CAPS OFF-Sets the names of all character columns to lowercase
- CAPS ASIS-Resets the case of column names to match the case of the data you entered in those columns

Consider the following when issuing the CAPS command:

- CA OPS/MVS does not change the case of data in columns containing unmodified data, no matter which version of the CAPS command you issue.
- CA OPS/MVS always translates hexadecimal data to uppercase.
- CAPS ASIS mode is in effect when you first enter an editing session.

Change

Replaces one text string with another:

- The command C *string1 string2* substitutes *string2* for this occurrence of *string1*.
- To search forward and change the next occurrence of *string1*, issue the command C *string1 string2* NEXT.
- To search backward and change the previous occurrence of *string1*, issue the command C *string1 string2* PREV.
- To change all occurrences of *string1*, issue the command C *string1 string2* ALL.

Note: If you substitute an asterisk (*) for one or both of the strings in any of these CHANGE commands, the table editor uses the string value or values specified on the previous CHANGE command.

DOWN

Scrolls down toward the bottom of the table.

END

Saves changes and exits this editing session.

FIND

Finds a specified text string and places the cursor on that string:

- To search forward for the string, issue the command F *string* NEXT.

  Note: The table editor searches forward by default.

- To search backward for the string, issue the command F *string* PREV.

- To search for the string specified on the last FIND command, issue the command F *.

  To change the direction of a search (but search for the same string), issue the command F * NEXT or F * PREV.

LEFT

Scrolls toward the left side of the column definitions.

RCHANGE

Repeats the last Change command issued.

RESet

Cancels all pending line commands and removes all SQL error messages from your screen.

RFIND

Repeats the last Find command issued.

RIGHT

Scrolls toward the right side of the column definitions.

UP

Scrolls toward the top of the table definition.

Line Commands for Editing Table Structure

To edit column definitions for a new table, enter these commands in the line number field, on the line you want to edit:

A

Places the copied or moved data in the table after this column.

B

Places the copied or moved data in the table before this column.

C

Copies this column definition.

CC

Uses this command to select a range of columns for copying. Enter CC beside the first line and the last line of the range.

D

Deletes this column.

DD

Uses this command to select a range of columns to delete. Enter DD beside the first line and the last line of the range.

I

Inserts a column.

M

Moves this column.

MM

Uses this command to select a range of columns to move. Enter MM beside the first line and the last line of the range.

R

Repeats this column.

RR

Uses this command to select a range of columns to repeat. Enter RR beside the first line and the last line of the range.

TJ*nnnn*

Uses this command to join text, shift it to the left, delete a word, or delete text to the right of the cursor.

TS*nnnn*

Uses this command to split text or to shift it to the right.

To begin editing a new table, issue I line commands to insert columns and type in the required data for each column. After you have inserted some columns, you can issue other line commands to manipulate those columns (for example, to copy or move columns).

Issuing the TJ and TS Line Commands

The TJ (Text Join) and TS (Text Split) line commands allow you to shift data easily when you insert or delete data in columns. For example, if you delete data from a column using the delete key for the ISPF/PDF editor, characters in that column that are beyond the right side your screen do not move into the space that data vacated. However, if you use the TJ command to delete characters, the characters in that column beyond the right border of your screen shift left.

The TJ line command works as follows:

- It determines where the cursor is.
- If the cursor is on a non-blank character, it deletes all data between the current cursor position and the next blank character.
- It deletes all blank characters until it finds the next non-blank character.

Suppose that you are editing a column called CURRENT_STATE, which contains these characters:

```
REGION IS DOWN
```

You can use the TJ line command to delete text from this column and join the characters to the left and right of the deleted text. Which characters are deleted depends on where the cursor is when you issue the TJ command.

For example, the following table shows you how different cursor positions affect the outcome of a TJ command. The underscore represents the cursor position. On the second line of the table, the TJ command deletes the word IS because the cursor is on the first character of that word.

Column Contents	Result of TJ Command
REGION_IS DOWN	REGIONIS DOWN
REGION IS DOWN	REGION DOWN
REGION IS DOWN	REGION IDOWN

If you insert characters into a column using the insert key for the ISPF/PDF editor, characters to the right of the inserted characters do not shift beyond the right border of your screen. However, if you use the TS command to insert characters, CA OPS/MVS shifts the existing characters as far right as necessary to make space for the new characters. For example, the following table shows you how different cursor positions affect the outcome of a TS command. The underscore represents the cursor position.

Column Contents	Result of TJ Command
REGION_IS DOWN	REGION IS DOWN
REGION IS DOWN	REGION IS DOWN
REGION IS DOWN	REGION I S DOWN

Usually, when you issue the TJ and TS commands, the table editor shifts data in a column as far right as possible without deleting any non-blank characters. However, to shift data past the end of a column and delete the data, you can specify an optional value (a number from 1 to 9999) with TJ or TS.

You can also use the optional number value when you want to delete or insert a certain number of characters to the right of (and including) the current cursor position. For example, if you issue the command TJ6, the table editor deletes the character in the current cursor position and the next five characters to the right of the cursor.

The following table shows you the results of issuing the TJ command with numeric values:

Command	Cursor Position	Result of TJ Command
TJ4	REGION_IS DOWN	REGIONDOWN
TJ9999	REGION_IS DOWN	REGION (all characters to the left of the cursor are deleted)

The following table shows you the results of issuing the TS command with numeric values.

Note: The bar character (|) represents the end of the column.

Command	Cursor Position	Result of TS Command
TS2	REGION IS DOWN |	REGION I S DOWN|
TS99	REGION IS DOWN |	REGION I |

Both the TJ and TS commands affect only the data in the column where the cursor currently rests. Data in other columns does not change position. For example, suppose that you issue the command TJ99 but the current column contains only 12 characters. The TJ command deletes the character in the current cursor position and all characters to the right of the cursor **in the current column**. However, any characters in columns to the right of the current column stay where they are.

Edit the Contents of an Existing Table

To edit the data in an existing table, use the Table Data Editor panel, shown here:

```
Table Data Editor --------------- TBLNAME ---------------- COLUMNS 00001 00072
Command ===>                                                  Scroll ===> PAGE
COL--> TABLE_ID NAME              TABLE_NAME         COL#   DATA_TYPE DATA_OFF
****** ************************** TOP OF DATA ********************************
000001 0001     AVMAN             TABLE              12     0300      62
000002 0001     COLUMN_NUMBER     TABLE              5      0500      28
000003 0001     CREATE_TIME       TABLE              11     0400      58
000004 0001     ID                TABLE              2      0100      20
000005 0001     NAME              TABLE              1      01A0      2
```

This example shows you a relational table. If the table contains more columns than fit on the screen, you can scroll left or right to see the rest of the columns.

Primary Commands for Editing Table Data

The primary commands available to edit table data resemble those used to edit table structure, except that the commands operate on actual data instead of column definitions.

The Table Data Editor panel also has an extended set of primary commands. If a command is shown in all uppercase letters, you must enter the full command name.

CANcel

Exits this table editing session without saving changes, or cancels a request to free a table.

Change

Replaces one text string with another:

- The command C *string1 string2* substitutes *string2* for this occurrence of *string1*.
- To search forward and change the next occurrence of *string1*, issue the command C *string1 string2* NEXT.
- To search backward and change the previous occurrence of *string1*, issue the command C *string1 string2* PREV.
- To change *all* occurrences of *string1*, issue the command C *string1 string2* ALL.

Note: If you substitute an asterisk (*) for one or both of the strings in any of these CHANGE commands, the table editor uses the string value or values specified on the previous CHANGE command.

COLUMN

Scrolls the display until the column name specified, *colname,* is at the far left of the display area. For example, issuing the command COLUMN PRE positions the first column name that begins with the characters PRE at the far left of the scrollable display area.

COPY [*system*>]*tablename*

Copies the named table into this edit session. You must specify the A line command or the B line command with the COPY command.

DOWN

Scrolls toward the bottom of the table.

Note: This command does not work if there is a pending line command or an SQL error.

END

Saves changes and exits this editing session.

Find

Finds a specified text string and places the cursor on that string:

- To search forward for the string, issue the command F string NEXT.

  Note: The table editor searches forward by default.
- To search backward for the string, issue the command F string PREV.
- To search for the string specified on the last FIND command, issue the command F *.

  To change the direction of a search (but search for the same string), issue the command F * NEXT or F * PREV.

KEYFIX

Adds a non-scrollable protected area of size *nn* characters *nn*/ON/OFF/? containing as much of the primary key value of the table as will fit in the area. Trailing blanks are removed from the key values. Multiple key values are separated by a single blank. A minimum of 19 characters is reserved for a scrollable column area, regardless of the fixed size specified. If the actual key length is less than the amount specified, the actual size will be substituted.

- KEYFIX ON-sets the display area size to the actual key size
- KEYFIX OFF, KEYFIX0, or KEYFIX-removes the key display area from the panel and restores the maximum size scrollable area to KEYFIX 0

The last KEYFIX command is stored in the ISPF profile of the user and is used subsequently to set the initial KEYFIX for all relational displays.

LEFT

Scrolls to the left side of the row.

RCHANGE

Repeats the last Change command issued.

REFRESH

Fetches and displays a new copy of the table data currently being edited. This command works only if the table data has not changed since you last saved it. If a rule or another user has altered the table, the REFRESH command produces an error message.

REPLACE [*system*>]*tablename*

Replaces the named table with the data currently being edited. You must specify one of these line commands with the REPLACE command: C, CC, M, or MM.

RESet

Cancels all pending line commands.

RFIND

Repeats the last Find command issued.

RIGHT

Scrolls toward the right end of the row.

SAVE

Saves the data without ending this table editing session.

UP

Scrolls toward the top of the table.

Note: This command does not work if there is a pending line command or an SQL error.

If an SQL error involving the whole table occurs when you issue the END, SAVE, or REPLACE command, an error message appears at line 3 of the Edit Table Display panel. You can delete this message only by issuing the RESET command.

If an SQL error involving only one line of a table occurs when you issue the END, SAVE, or REPLACE command, CA OPS/MVS displays an error number (in the format <*nnnn*>) in the line number field for the affected line. CA OPS/MVS also displays the text of the bad row and the text of the associated error message. To delete the associated error message, either issue the RESET command or type blanks over the error number in the line field.

Line Commands for Editing Table Data

From the Table Data Editor panel, you can issue the same line commands that are available in the Table Structure Editor panel.

More information:

Line Commands for Editing Table Structure (see page 454)

Edit a Table on Another System

When CA OPS/MVS copies are running on two or more systems and communicating through the CA OPS/MVS MSF or CAICCI cross-platform communications services, you can edit a relational table residing on a remote system using the table editor on the local system.

To edit a table on a remote system, you specify both the system name and the table name on the RDF Table Editor Primary Panel. Separate the system name and the table name with a greater-than symbol (>) instead of a blank, like this:

systemname>tablename

You can specify the ? and * characters instead of a specific system name. The following list shows how to use these characters as wildcards:

- To select one of the tables defined on the current system:

  Enter the name of that table.

- To select a table from a list of tables on the current system:

  Enter an asterisk (*), or leave the Name field blank. In response, you see a panel listing available tables.

- To select a specific table on another system:

  Enter the name of the system followed by the greater-than symbol (>) and the table name, as shown here:

  sysname>tablename

- To select a list of all systems having defined tables:

  Enter the characters ?>* or ?>(blank)

  In response, you see a list of systems running CA OPS/MVS. To select one or more systems, type *S* beside the system name or names.

  Note: You cannot select systems whose status is not active.

  Result-You see a list of all relational tables on the selected systems.

- To select a list of all tables named *tablename:*

  Enter the following text: ?>*tablename*

  This text causes the table editor to display a list of systems running CA OPS/MVS. After you choose one or more systems, the table editor displays a list of all tables on the selected systems that have the name *tablename*.

- To select a list of all tables on all systems

  Enter any of the following character sets:

  (blank)>(blank)

  (blank)>*

*>(blank)

>

- To select a list of all tables named *tablename* on all systems:

  Enter one of the following:

  (blank)>*tablename*

  *>*tablename*

  This text displays a list of all tables with the specified name without displaying a list of systems.

- To select a list of all tables on the specified system:

  Enter one of the following:

  sysname>(blank)

  sysname>*

End a Table Editing Session

To end a table editing session, saving any changes that you made to a relational table, press PF3 or type END at the command line and press Enter. CA OPS/MVS updates the table automatically and returns you to the RDF Table Editor Primary Panel or the Table List panel. To return from this panel to the main OPSVIEW panel, press PF3 or type END again.

To end a table editing session without saving your changes, type CANCEL at the command line and press Enter. Your session ends, leaving your table unchanged.

Chapter 13: External Product Interface

This section contains the following topics:

Overview

The External Product Interface, or EPI, allows CA OPS/MVS systems running on processors that use VTAM to interface with any VTAM application that supports IBM 3270 (SLU2) type virtual terminals. The EPI appears to VTAM as a real 3270 terminal that can emulate any number of 3270 type virtual terminals connected to any number of VTAM applications.

Note: The EPI supports virtual IBM 3278 models 2, 3, and 4 only. Extended attributes and programmed symbols are not supported in the initial release. For information about the OPSVIEW panels that control EPI virtual terminals, see the *OPSVIEW User Guide*.

The OPS/REXX programming language enables you to automate operator actions and tasks; REXX programs can drive 327X sessions with other VTAM applications. Specifically, the EPI enables you to create REXX programs to:

- Log on to a VTAM application.
- Enter commands and data from a virtual terminal keyboard.
- Read data from the virtual terminal screen.
- Log off from VTAM.

In addition, the EPI gives you a way to share a session. For example, a single session with OMEGAMON (running in VTAM mode) can be shared by multiple OPS/REXX programs without requiring each program to go through the logon/transaction/logoff sequence. Through a mechanism similar to ENQ/DEQ logic, used to share data sets between z/OS tasks, the EPI enables you to share VTAM sessions between OPS/REXX programs.

How the EPI Manages VTAM Applications

Together with the Automated Operations Facility (AOF), you can use the EPI to interact with VTAM applications in the following specific ways:

- A message is issued from a message rule.
- A command is issued.
- An OMEGAMON event occurs from an OMEGAMON rule.
- A user-initiated event occurs from a request rule.
- A time of day event occurs from a TOD rule.

Components List

The EPI consists of the following components:

- A host command environment for the OPS/REXX language (ADDRESS EPI)
- A control panel (option 4.10 of OPSVIEW) that allows you to control and monitor the EPI facility
- An extension to AOF that supports screen rules
- Sample OPS/REXX programs (such as CA7MVS, OMMVS, OPEPCM, and OPEPDFAL located in data set SYS1.OPS.CCLXEXEC) that demonstrate the use of the EPI

EPI Terminology

The EPI is not complicated, but to use it you should understand the following terms:

VTAM application

A VTAM application is an application known to VTAM by its VTAM application ID.

External product

An external product is an application or group of applications external to CA OPS/MVS with which the EPI can communicate. Usually each external product uses a single VTAM application ID, but this is not always true. A single product can use multiple VTAM application IDs, or a single VTAM application ID can give access to multiple external products.

Note: The EPI communicates only with external products that support IBM 327X type virtual terminals under VTAM. The EPI establishes sessions only with applications that are in the same VTAM domain as CA OPS/MVS or which are accessible as resources from that domain.

Virtual terminal

A virtual terminal is anything that looks like a physical terminal to VTAM. In this sense, the EPI emulates a physical terminal.

Disabled virtual terminal

A disabled terminal is not connected to VTAM (no ACB is open for it) and cannot communicate or establish a session with any external product. Disabling a virtual terminal is the equivalent of shutting the power off on a physical terminal.

Enabled virtual terminal

An enabled terminal is connected to VTAM (an ACB is open for it). An enabled virtual terminal can establish a session with any external product, or can be in session with an external product.

Active virtual terminal

A virtual terminal is active when it is enabled and has established a session with an external product.

Note: After the EPI establishes a VTAM session with an external product, the external product can present a prompt sequence for signon information. While the EPI shows that a session has been established at that point, the external product cannot consider the session established.

Session

An EPI session is a logical connection through a VTAM session between CA OPS/MVS and an external product. The EPI communicates only with an external product if a session exists between the two products.

Install the EPI

You cannot start the EPI automatically at product initialization and then leave it alone; this is because the EPI communicates with other VTAM applications that may be active when the EPI (CA OPS/MVS) starts. The EPI depends on VTAM availability, unlike the rest of CA OPS/MVS. The following discussion assumes that you operate the EPI manually. However, to automate any EPI operation task that you perform regularly and manually, you should make the maximum use of the CA OPS/MVS AOF component.

Take the following steps to prepare to use the EPI:

1. Add and activate virtual terminal definitions to VTAM.

   This step adds APPL statements to the VTAM definition library and activates these definitions in VTAM.

2. Define virtual terminals to the EPI.

   This step notifies the EPI of the virtual terminal names it should use, their characteristics, and which external products they will be used to logon to (see the EPI DEFINE command).

3. Enable the virtual terminal or terminals.

   This step connects the virtual terminals to VTAM (see the EPI ENABLE command).

4. Log the virtual terminal onto an External Product.

   This step establishes the connection between the virtual terminal and an external product (see the EPI LOGON command).

These steps are discussed in detail in the following sections.

Define Virtual Terminals to VTAM

You can install and initialize most of the EPI. However, virtual terminal APPLs must be written to VTAM when CA OPS/MVS is installed or must be active before you define the virtual terminals to the EPI. If the terminals already have VTAM IDs, continue to define the terminals to the EPI.

Note: For details about writing APPLs, see the *Installation Guide*.

Define Virtual Terminals to the EPI

Once CA OPS/MVS has been initialized, you must define the virtual terminals to the EPI.

To define virtual terminals to the EPI, use one of the following methods:

- Use the OPEPDFAL routine provided in the REXX distribution library to define all terminals under a given VTAM major node.
- Use the DEFALL command under option 4.10 of OPSVIEW to define all terminals under a given VTAM major node.
- Write an OPS/REXX program to issue an EPI DEFINE command for each terminal.
- Use option 4.10 of OPSVIEW to manually define each terminal.

OPEPDFAL Routine—Define All Terminals to the EPI

The OPS/REXX program OPEPDFAL can be called from any REXX program to define all terminals to the EPI under a given VTAM major node.

This routine takes one argument, a character string composed as follows:

```
RETCODE = OPEPDFAL('majnode keywords')
```

majnode

The name of the VTAM major node (see the VBUILD statement discussed previously).

keywords

Any keywords allowed on the EPI DEFINE command.

Example: OPEPDFAL Routine

This example defines all APPLIDs under VTAM major node name OPSSAPPL as virtual terminal names under the EPI, using a VTAM logmode table entry named T3278M3:

```
RETCODE = OPEPDFAL('OPSSAPPL LOGMODE(T3278M3)')
```

DEFALL Command—Define All Terminals in OPSVIEW

While viewing the terminal list under option 4.10 of OPSVIEW, you can issue the DEFALL command to define all terminals under a given VTAM major node name.

This command has the following syntax:

```
DEFALL majnode keywords
```

majnode

The name of the VTAM major node (see the VBUILD statement discussed previously).

keywords

Any keywords allowed on the EPI DEFINE command.

Example: DEFALL Command

This example will define all APPLIDs under VTAM major node name OPSSAPPL as virtual terminal names under the EPI:

```
DEFALL OPSSAPPL LOGMODE(T3278M3)
```

Write an OPS/REXX Program

You can write a small OPS/REXX program to define each virtual terminal to the EPI.

For example, this defines three virtual terminals to the EPI:

```
ADDRESS "EPI"
   'DEFINE OPSS0001 LOGMODE(T3278M3)'
   'DEFINE OPSS0002 LOGMODE(T3278M3)'
   'DEFINE OPSS0003 LOGMODE(T3278M3)'
```

Note: For detailed information about the EPI DEFINE command, see the *Command and Function Reference*.

Use the EPI DEFINE Command

Issue an EPI DEFINE command for each virtual terminal you intend to use. You can issue the command automatically using the AOF, or from any OPS/REXX program.

Enable Virtual Terminals

After you have defined a virtual terminal to the EPI, it should be enabled.

To enable virtual terminals

Issue the following EPI ENABLE command automatically using the AOF, or from any OPS/REXX program:

`ENABLE` *keywords*

When one of these commands is issued, the EPI issues a VTAM OPEN ACB request.

Option 4.10 of OPSVIEW provides a panel to accomplish this same task.

Note: For descriptions of the keywords for the ENABLE command, see the *Command and Function Reference*.

Log a Virtual Terminal onto an External Product

Once you enable a virtual terminal through the EPI ENABLE command, you can log the virtual terminal onto an external product.

To log a virtual terminal onto an external product

Issue the following EPI LOGON command automatically using the AOF, or from any OPS/REXX program:

`LOGON` *keywords*

When this command is issued, the EPI issues a VTAM REQSESS request.

For descriptions of the keywords for the LOGON command, see the *Command and Function Reference*.

Define and Activate EPI Sessions

This procedure builds an EPI configuration consisting of three virtual terminals.

To define and activate EPI sessions

1. Execute the following REXX program once CA OPS/MVS has completed initialization:

   ```
   EPI DEFINE OMTERM1,APPLID(OMVTAM) LOGONPARM()
   EPI DEFINE TSOTERM1 APPLID(TSO) LOGONPARM('TSOID1')
   EPI ENABLE ALL
   EPI LOGON ALL
   ```

   The above example defines three virtual terminals, enables all of them, and logs all of them to their respective external products.

Example: Define and activate EPI sessions

The following ADDRESS EPI example does the following:

- Defines a specific terminal name (CA7TERM1)
- Enables all terminals
- Logs the CA7TERM1 on to its external product
- Enables and logs on all other terminals but does not assign them to a specific external product

```
ADDRESS EPI
 DEFINE CA7TERM1,APPLID(CA7)
 ENABLE ALL
 LOGON CA7TERM1
```

External Products Acquiring Virtual Terminals

As soon as the EPI ENABLE command is issued for a virtual terminal, an external product can acquire that terminal. The external product initializes and activates the terminal instead of the terminal being activated by an EPI LOGON command.

The ACCEPT/REJECT setting of a virtual terminal determines whether the EPI allows a particular virtual terminal to be acquired by any external product. The default value of ACCEPT, indicates that any external product will be allowed to acquire the virtual terminal. The operand REJECT prevents any attempt to acquire the terminal.

Display Virtual Terminal and EPI Session Information

You can display the virtual terminals defined to the EPI and their status, by issuing the EPI LIST command from a REXX program:

```
ADDRESS EPI "LIST keywords"
```

Note: For descriptions of the keywords for the LIST command, see the *Command and Function Reference*.

Display Virtual Terminal Status

A virtual terminal can have the following status:

DISABLED

The virtual terminal has been defined but is not running a session. An ENABLE and LOGON command are necessary to start a session.

ENABLED

The virtual terminal has been defined and enabled, but no session is active on it. A LOGON command or an external product acquire request is necessary to start a session.

ACTIVE

The virtual terminal has been defined and enabled and a session with an external product is active. A DISABLE command is necessary to disable the virtual terminal.

RETRYING

The virtual terminal has been defined and enabled, but a LOGON command did not complete successfully, or a session was in progress and terminated prematurely and RETRY mode was in effect. The EPI will attempt to reestablish the session up to the maximum number of retries allowed. A DISABLE command takes the virtual terminal out of service. A LOGON command reestablishes the session before the next retry attempt.

EPI LIST Command

The following is a sample display of the information returned by the EPI LIST ALL command:

```
                                VTAM      VTAM    === RETRY ===     VTAM
TERMINAL USERNAME  STATUS    APPLNAME  LOGMODE   SECS MAX  NOW  RTNCD FDBK2
-------- --------  ------    --------  --------  ---  ---  ---  ----- -----
OMTERM1            ACTIVE    OMVTAM    T3278M2    30   30   0   X'00' X'00'
OMTERM2            ENABLED   OMCICS    T3278M3    30   30   0   X'10' X'01'
CA7TERM            RETRYING  CA7       T3278M4    30    4   2   X'10' X'00'
OPS00001           ACTIVE    TSOTULD   T3278M3    NO RETRY      X'00' X'00'
OPS00002           ACTIVE    TSOIPD0   T3278M2    NO RETRY      X'00' X'00'
```

This display shows the following information:

- Five virtual terminals have been defined.
- The first virtual terminal (OMTERM1) is enabled and in session with an external product whose VTAM application name is OMVTAM.
- The second virtual terminal (OMTERM2) is enabled but not in session with any external product.
- An EPI LOGON command at this point would attempt to establish a session with an external product whose VTAM application name is OMCICS.
- The third virtual terminal (CA7TERM) is enabled and has a problem establishing a session to an external product whose VTAM name is CA7.
- This virtual terminal was defined with RETRY mode for a maximum of four retries. The EPI is retrying to establish the session. So far the EPI has retried twice to establish the session.
- An attempt to establish a session failed with a VTAM RTNCD of 10 and FDBK code of 0. For a detailed explanation of these codes, see your *VTAM Programming* guide.

Note: Like any part of the z/OS environment, you can operate the EPI with the AOF. If the NORETRY and RETRY options supported by the EPI are not sufficient to establish EPI sessions and keep them active in your environment, you can program more sophisticated procedures using the same AOF environment control facilities.

Following is sample output from the EPI LIST OMTERM1 command:

```
                                VTAM      VTAM   === RETRY ===     VTAM
TERMINAL USERNAME  STATUS   APPLNAME  LOGMODE  SECS MAX  NOW  RTNCD FDBK2
-------- --------  ------   --------  -------- ---  ---  ---  ----- -----
OMTERM1            ACTIVE   OMVTAM    T3278M2   30   30   0   X'00' X'00'
LOGONPARM =
ROWS = 24
COLUMNS = 80
VTAM SENSE = 00000000
MAX RCVE RU SIZE = 512
MAX SEND RU SIZE = 512
ACCEPT
NORETRY
TRACE&=&OFF
ENQ USER=TSOUSER1 ASCB=X'00F31C00' TCB=X'007E83C0'
ENQ USER=TSOUSER2 ASCB=X'00F32C00' TCB=X'008E8440'
```

Shut Down the EPI

To shut down the EPI facility, do one of the following:

- Log off each active virtual terminal.
- Disable all virtual terminals.

The procedure to log off a virtual terminal that is in session with an external product depends on the external product. For example, to log off from TSO, you must get to TSO READY mode, and then type in the TSO LOGOFF command at the virtual terminal.

A quick shutdown can be accomplished by merely disabling all virtual terminals with the EPI command:

```
DISABLE ALL
```

This action takes all virtual terminals out of service.

Note: The DISABLE ALL command should be used with caution. One or more of your external products can require a proper logoff sequence before a virtual terminal is disabled.

Disable Virtual Terminals

When you disable a virtual terminal, it can no longer communicate between the external product the terminal was in session with and the EPI.

To disable an EPI virtual terminal and kill any session in progress

1. Issue the following EPI DISABLE command from a TSO virtual terminal or from a console logged on to the CA OPS/MVS ECF facility:

   `DISABLE` *keywords*

   Communications are cut between the external product the terminal that was in session with and the EPI. Any commands in progress over the EPI session are aborted and a message is issued that indicates that the virtual terminal was disabled.

For descriptions of the keywords for the DISABLE command, see the *Command and Function Reference*.

Delete Virtual Terminal Definitions

You can delete the definition of a virtual terminal (previously added with an EPI DEFINE command).

Note: You cannot delete a definition for a virtual terminal on which a session is active until you have disabled the virtual terminal using EPI DISABLE.

To delete virtual terminal definitions

Issue the following EPI DELETE command:

`DELETE` *keyword*

You need to issue this command only if you want to change the name of a virtual terminal. To change any other virtual terminal characteristic, use the EPI DEFINE command.

For descriptions of the keywords for the DELETE command, see the *Command and Function Reference*.

ops--Use OPS/REXX to Drive EPI Virtual Terminals

Once an EPI virtual terminal has logged onto an external product, you can use the OPS/REXX ADDRESS EPI environment to drive the session. This environment provides a set of commands that you can use to read information from the virtual 3270 screen and enter information from the virtual 3270 keyboard.

Issue ADDRESS EPI Host Commands

To issue commands to the EPI from an OPS/REXX program, you must code an ADDRESS statement of the form:

```
ADDRESS EPI
```

For clarity, the word EPI can be specified as the first word of each host command, but this is not required. The following host commands are functionally equivalent:

```
LIST TERM1
'LIST' TERM1
'LIST TERM1'
EPI LIST TERM1
EPI 'LIST' TERM1
```

The first and simplest form is the one that is used in all of the examples.

General Syntax Rules of ADDRESS EPI

ADDRESS EPI command syntax, the same as other REXX statements, is as follows:

- The EPI supports words with a generated length of up to 64 characters.
- All words can be in uppercase, lowercase, or mixed case. If they convey literal information, all words must be placed in quotes.
- Symbols, integers, quoted strings, hex strings, and comments are all supported.
- Numbers are restricted to integers.

Output from ADDRESS EPI Host Commands

Output from the EPI host commands is always returned in the REXX external data queue.

The commands that can generate multiple lines of output are:

- LIST
- HELP
- PEEK
- RDSCREEN
- TYPETEST

ADDRESS EPI Return Codes

Subsequent to execution of an ADDRESS EPI command in a REXX program, the variable RC in the program will contain the execution return code.

<0

Host command failure. An ABEND occurred while processing the EPI command.

0

OK. Command executed successfully.

>0

Host command errors.

4

Warning message was issued.

8

Command timeout error. Not all responses were received. The command took too long to complete.

12

Command failed. An error message was issued.

16

EPI command syntax error.

20

Subsystem is not active. CA OPS/MVS is not running.

24

Incompatible subsystem version.

28

SENDMG failed.

32

Authorization exit rejected EPI command.

36

User exit abended.

ADDRESS EPI Output Message Identification

Each line stored in the external data queue contains a message identification section in the first eight positions.

```
OPSnnnna
     OPS = fixed text
    nnnn = four numeric characters
       a = the message i.d. suffix
           I - information
           A - action
           W - warning
           E - error
           S - severe error
           U - unrecoverable error
           H - hardcopy
Range 0075 thru 0076 is reserved for messages from OPINEP
Range 3550 thru 3589 is reserved for messages from OPEPEX
Range 4380 thru 4399 is reserved for messages from OPRXEP
```

Syntax of Selected ADDRESS EPI Words

The following describes the syntax of selected EPI words.

- The following syntax is the name of the virtual terminal:

  ```
  termname
  ```

  The *termname* can exist in one of the following three different forms:

  - A single character * representing the current virtual terminal defined with the SETTERM command.
  - The name of the virtual terminal as defined to VTAM (the name from the APPL statement in the VTAM definition list.); this name can be from one to eight characters long
  - The name assigned by the SETUNAME command

- The following syntax is the screen row number:

  ```
  Row
  ```

  The top row is row 1. Operator information area (OIA) is row 0 (even though it is physically on the bottom of the screen).

- The following syntax is the screen column number.

  ```
  Col
  ```

  The leftmost column of any screen row is col 1.

- The following syntax is the length of the data to be displayed:

  ```
  Length
  ```

- The following syntax indicates that the string is either ON or OFF:

  ```
  ON|OFF
  ```

  Any string starting with Y (that is, YES) is the same as ON. Any string starting with N (that is, NO) is the same as OFF.

REXX Use of the Virtual Terminal Temporary Ownership Mechanism

For some operations, you should consider a virtual terminal as a serially reusable resource. For example, when two REXX programs use the same virtual terminal to issue a sequence of OMEGAMON commands, problems occur if both programs try to issue the commands at the same time. The mechanism of temporary ownership exists to alleviate this type of problem. Temporary ownership basically locks the terminal to a specific program until it is released. Ownership is either of the following:

- implicit
- explicit

Implicit

As the REXX program is being run, virtual terminal ownership is implicitly obtained whenever the program issues a command requiring the virtual terminal. Ownership is released as soon as the command has executed.

Explicit

A REXX program can first explicitly issue the ENQ or BIND command to establish virtual terminal ownership, and then use the virtual terminal for a series of commands, and finally issue the DEQ or UNBIND command to release ownership.

EPI Host Command Descriptions

The next sections describe the EPI host commands. They are:

- Special host commands
- Virtual terminal host commands
- Other host commands

Special EPI Host Commands

The EPI host commands in this section affect the output format of all other host commands.

At the start of execution of any REXX program or AOF rule, the following defaults are in effect for the program.

MSGID ON

This default causes the message IDs to be prefixed to each output line returned in the EPI external data queue.

SUBATTR 64 OFF

This default returns all attribute bytes as values between X'20' and X'3F' (no translation).

SUBUNPT : ON

This default returns all unprintable characters as colons.

Detailed explanations of the special host commands are presented next.

MSGID [ON|OFF]

This command turns the OPS message prefix in output lines ON or OFF. Output lines with a message prefix are nine characters longer than those without one. Message prefixes are comprised of an eight-byte message ID and a trailing blank.

Default: ON

Note: The program return code indicates the value of the setting before the command was issued. A subroutine can be used to temporarily alter the value, and then restore the environment to its former state. For example:

```
mysubroutine:
   MSGID OFF
   saved_value = rc
   ... ... ...
   IF saved_value = 1 then
     MSGID ON
   Else
     MSGID OFF
   return
```

Following are the return codes:

0

Normal completion, former setting was MSGID OFF

1

Normal completion, former setting was MSGID ON

SUBATTR [sub_char] [ON|OFF]

This command affects the way the RDSCREEN and RDSCRROW commands return information.

If you specify a single character operand (*sub_char*), the SUBATTR command sets the substitute character to be used for all host attribute byte characters. The argument can be a single character or a numeric value in the range 0 through 255 corresponding to the EBCDIC value of the desired character. The program return code indicates the value the setting before the command was issued. A subroutine can be used to temporarily alter the value, and then restore the environment to its former state. If you specify OFF, attribute characters are not translated.

If you specify ON, attribute characters are translated. This reverses the effect of a previously issued SUBATTR OFF command.

Default: OFF

Note: The SUBATTR command only affects the output returned from the RDSCREEN and RDSCRROW commands. Attribute bytes are not actually modified on the screen of the virtual terminal by this command.

The following example sets the value to a percent sign:

```
SUBATTR %
```

Return codes represent the EBCDIC equivalents of the previous SUBATTR value.

SUBUNPT [sub_char] [ON|OFF]

Use this command to affect the way the RDSCREEN and RDSCRROW commands return information.

If you specify a single character operand (*sub_char*), the SUBUNPT command sets the substitute character to be used for unprintable characters. The argument can be a single character or a numeric value in the range of 0 through 255 that corresponds to an EBCDIC value. The program return code indicates the setting value before the command was issued. A subroutine can be used to temporarily alter the value, and then restore the environment to its former state.

If you specify OFF, unprintable characters are not translated. If you specify ON, unprintable characters are translated. This reverses the effect of a previously issued SUBUNPT OFF command.

Default: OFF

The SUBUNPT command only affects the output returned from the RDSCREEN and RDSCRROW commands. Unprintable characters are not actually modified on the screen of the virtual terminal by this command.

The following example sets the value of SUBUNPT to a slash:

```
SUBUNPT / ON
```

Return codes represent the EBCDIC equivalents of the previous SUBUNPT value.

SUBSYS ssid

Use this command to direct all subsequent EPI host commands from the current OPS/REXX program to an EPI other than that using the default CA OPS/MVS z/OS SSID of OPSS. You will only need this command if you are running multiple copies of CA OPS/MVS at once and are, therefore, using multiple SSIDs.

Note: This command is not allowed in any AOF rules with the exception of request rules.

ssid

This operand is the four-character z/OS subsystem identifier (SSID) of the copy of CA OPS/MVS. All subsequent EPI host commands from the current OPS/REXX program are directed to this subsystem until another EPI SUBSYS command is encountered.

The following example directs all subsequent EPI commands to subsystem OPSM:

```
SUBSYS OPSM
```

TIMEOUT seconds

Use this command to change the default timeout value. Usually, the EPI waits for 60 seconds before timing out. When a timeout occurs, the message EPI COMMAND TIMED OUT BEFORE ALL RESPONSES RECEIVED is returned in the external data queue. If you want to wait a longer or shorter interval for subsequent EPI commands to time out, use the TIMEOUT command to specify a different value (in seconds).

Seconds

The number of seconds (from 1 to 86400) to wait before subsequent EPI commands will time out. The following example will change the timeout value for all subsequent EPI commands to two minutes (120 seconds):

```
TIMEOUT 120
```

EPI Host Command Descriptions for Virtual Terminals

Use the following EPI host commands to control virtual terminals.

CHANGE [termname|ALL|*] keywords

Use the CHANGE command to change the attributes of a previously defined terminal. Any of the parameters used to define a terminal can be changed later with a CHANGE command.

```
ADDRESS EPI "CHANGE keywords"
```

For descriptions of the keywords for the CHANGE command, see the *Command and Function Reference*.

DEBUG [ON|OFF]

Turns VTAM exit debugging on or off. When DEBUG mode is on, the EPI will write a message to hardcopy every time a VTAM exit is entered.

Default: OFF

```
DEBUG ON
```

DEFINE termname keywords

This command is explained in full in a previous section.

```
DEFINE TERM1 APPLID(OMVTAM) LOGMODE(T3278M2)
Return code:
 0 - Terminal now defined.
 4 - ALL cannot be used as a terminal name.
12 - DEFINE command failed. An error message was issued:
     Terminal name is blank.
     Username or terminal name already defined.
```

DELETE termname

This command is explained in full in a previous section.

```
DELETE TERM2
Return code:
 0 - Terminal(s) now deleted.
 4 - DELETE command was not executed. A warning message was issued:
     No disabled terminals were found to delete (DELETE ALL).
     Terminal name not found.
     Terminal is enabled.
12 - DELETE command failed. An error message was issued.
```

DEQ termname [FORCE]

Use this command to release ownership (dequeue) of the virtual terminal. If you specify the FORCE operand, it forces the current ownership to be released (even if the program current program does not have ownership). The FORCE operand should be used with extreme caution.

Note: This command is not allowed in any AOF rules with the exception of request rules.

```
DEQ TERM2
Return code:
 0 - DEQ completed successfully.
 4 - DEQ command was not executed. A warning message was issued:
     No enabled terminals were found to deq (DEQ ALL).
     Terminal name not found.
     Terminal is disabled.
12 - DEQ command failed. An error message was issued:
     No ENQ was found.
     Error posting next enq in chain.
```

DISABLE termname

Use this command to disable a virtual terminal. This is equivalent to placing the TEST/NORM switch to the TEST position. Following is an example of the command and its associated return codes:

```
DISABLE TERM1
Return code:
  0 - Normal completion.
  4 - No terminals disabled (DISABLE ALL).
    - Terminal is not defined.
    - Terminal is already disabled.
```

ENABLE termname

Use this command to enable a virtual terminal. This is equivalent to placing the TEST/NORM switch to the NORM position. Following is an example of the command and its associated return codes:

```
ENABLE *
Return code:
  0 - Normal completion.
  4 - No terminals enabled (ENABLE ALL).
    - Terminal is not defined.
    - Terminal is already enabled.
 12 - ENABLE failed. Error msg was issued.
```

ENQ termname [TEST|WAIT|NOWAIT]

Use this command to enqueue a virtual terminal or test for terminal ownership. Note that this command is not allowed in any AOF rules with the exception of request rules.

The operands have the following meanings:

WAIT

If you specify WAIT or no operand, control is not returned until the virtual terminal is owned.

TEST

If you specify TEST, control is returned immediately and the return code is set to indicate the ownership status.

NOWAIT

If you specify NOWAIT, ownership of the virtual terminal is assigned if it is immediately available; otherwise, no ownership is assigned.

Note: You must perform multiple ENQs in ascending order by terminal name.

If DEQ is not called after an ENQ, the virtual terminal is automatically dequeued when the REXX program ends.

```
ENQ TERM1
Return code:
  0 - Normal completion:
      For WAIT - ownership has been assigned.
      For TEST - terminal was available without wait.
      For NOWAIT - ownership has been assigned.
  4 - ENQ command not executed. A warning message was issued.
      No enabled terminals found to ENQ (ENQ ALL).
      Terminal is not defined.
      Terminal is disabled.
      You are already enqueued on terminal.
      You already have an ENQ pending for this terminal.
 12 - ENQ failed. An error message was issued.
      Terminal is already enqueued by another user.
      Conflict with another ENQ you issued previously.
      ENQ request area GETMAIN failed.
```

INQINPUT termname [WAIT|NOWAIT]

This host command is helpful before you use the TYPE command to ensure that the virtual keyboard is not locked (the Input Inhibited indicator is off). A TYPE command issued when the virtual keyboard is locked is usually rejected (the TYPE !RESET and TYPE !ATTN being the notable exceptions). If you specify neither WAIT nor NOWAIT, NOWAIT is assumed. When you specify NOWAIT, the current state of the input inhibit flag is returned immediately. When WAIT is coded, the issuing program will be placed into a wait until the external product turns off input inhibited.

Note: If you want to force the input inhibit light off, you can issue a TYPE command as follows:

```
TYPE termname !RESET
```

This resets the input inhibited indicator. It is possible, however, that the external product will immediately turn the indicator back on.

Note: This command is not allowed in any AOF rules with the exception of request rules.

An example of the command and its associated return codes follows:

```
INQINPUT *

Return code:
  0 - Input is not inhibited.
  1 - Input is inhibited.
      Most scancodes (except !RESET) will not work.
  4 - virtual terminal is not defined.
 12 - virtual terminal is not enabled.
    - INQINPUT command failed.
```

You can use the LIST host command to get a list of all or selected virtual terminals before you issue control commands.

LIST [termname|ALL|*]

The LIST command displays the status of virtual terminals. If an operand is omitted, all defined virtual terminals are displayed. If you specify ALL, then all virtual terminals are displayed. If you specify a *termname*, only the selected virtual terminal is displayed.

Note: This command runs synchronously in AOF rules and will return data in the external data queue.

```
LIST ALL
                              VTAM           === RETRY ===     VTAM
TERMNAME USERNAME  STATUS  APPLNAME LOGMODE SECS  MAX  NOW  RTNCD FDBK2
-------- --------  ------  -------- -------  ---  ---  ---  ----- -----
OMTERM3  OMEGAMON ACTIVE   OMVTAM   T3278M3   30   30   0   X'00' X'00'
OMTERM1  OMVTAM   ACTIVE   OMVTAM   T3278M3   30   30   0   X'00' X'00'
OMTERM2  OMCICS   ENABLED  OMVTAM   T3278M3   30   30   0   X'10' X'01'
CA7TERM  CA7      RETRYING CA7                30    4   2   X'10' X'00'
OPSS0001 TSO      ACTIVE   TSO                NO RETRY      X'00' X'00'
OPSS0002 TSO      ACTIVE   TSO                NO RETRY      X'00' X'00'
 Or
LIST OMTERM3
                              VTAM           === RETRY ===     VTAM
TERMNAME USERNAME  STATUS  APPLNAME LOGMODE SECS  MAX  NOW  RTNCD FDBK2
-------- --------  ------  -------- -------  ---  ---  ---  ----- -----
OMTERM3  OMEGAMON  ACTIVE  OMVTAM   T3278M3   30   30   0   X'00' X'00'
Return code:
 0 - Normal completion.
 4 - No terminals defined (LIST ALL).
   - Terminal is not defined.
```

LOGON termname [keywords]

This command is described in detail in a previous section.

```
MSGID OFF
LOGON TERM4
TERMINAL TERM4 LOGON ACCEPTED
Return code:
  0 - Normal completion.
  4 - No terminals logged on (LOGON ALL).
    - virtual terminal is not defined.
    - virtual terminal is disabled.
    - virtual terminal is already logged on.
 12 - LOGON command failed.
```

LOGOFF termname

This command disconnects the session between an external product and the virtual terminal. After the command completes, the virtual terminal is inactive, but enabled.

```
MSGID OFF
LOGOFF TERM4
TERMINAL TERM4 LOGGED OFF
Return code:
  0 - Normal completion.
  4 - No terminals logged off (LOGOFF ALL).
    - virtual terminal is not logged on.
    - virtual terminal is not defined.
 12 - LOGOFF command failed.
```

MVCURSOR termname row column

Moves the cursor to a specific row and column on the screen.

```
MVCURSOR TERM1 1 1
Return code:
 0 - Command completed successfully.
 4 - MVCURSOR command not executed. Warning message was issued:
     Terminal name is not defined.
     Terminal is disabled.
12 - Invalid row/column specified.
```

PEEK termname row col length

Use this command to display virtual terminal screen buffer contents. This command displays the raw buffer code in hexadecimal. Sixteen bytes are displayed in hexadecimal for each output line.

The typical use of this command is to interrogate 3278 host attribute bytes or to examine the Operator Information Area (Row 0). This buffer code is returned in EBCDIC format by the PEEK command. The ranges X'20' through X'3f' are attribute bytes. For an easier way to read the screen, see the RDSCRROW and RDSCREEN commands. The following example shows the beginning of the TSO READY message. The first 24 is a high-intensity unprotected attribute byte. The next five characters are READY in 3278 buffer code. The next 20 is a low-intensity unprotected attribute byte.

This command runs synchronously in AOF rules and will return data in the external data queue.

```
PEEK * 22 1 30
24 D9 C5 C1 C4 E8 20 00 00 00 00 00 00 00 00 00 *.READY.........*
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Return code:
  0 - Normal completion.
  4 - virtual terminal is not defined.
    - virtual terminal is not enabled.
 12 - Invalid row/column/length specification.
      The row or column is larger than the screen size
      or the length spans beyond the end of the screen.
```

POKE termname row col poketext

Use this command to modify virtual terminal screen buffer contents.

Note: Row zero (the Operator Information Area line) cannot be updated with POKE under the EPI.

WARNING! This host command can wreak havoc with the 3270 terminal protocols. It is intended solely for debugging purposes.

The operand Poketext is a REXX hex-string in quotes. The value is the actual 3278 buffer code. See the note about 3278 buffer code in the PEEK command above. The following example pokes the string READY at row 22, column 2:

```
POKE TERM4 22 2 "D9 C5 C1 C4 E8 20"x
  0 - Normal completion.
  4 - POKE command was not executed. A warning message was issued.
      Virtual terminal is not defined.
      Virtual terminal is disabled.
 12 - POKE command failed. An error message was issued.
      Invalid row/column/length specified for screen size.
```

RDCURSOR termname

This command returns cursor location. The location of the cursor is returned as two numbers, row, and column.

Note: This command runs synchronously in AOF rules and will return data in the external data queue.

```
MSGID ON
RDCURSOR TERM3
OPS3582I 23 1
Return code:
  0 - Normal completion.
  4 - RDCURSOR command was not executed. A warning message was issued.
      Terminal name is not defined.
      Terminal name is disabled.
```

RDSCREEN termname

This command returns the entire virtual terminal screen as multiple output lines. The first output line is the operator information area, which is the status line at the bottom of the display screen. The remaining lines are the contents of the screen starting with the top row.

Note: This command runs synchronously in AOF rules and will return data in the external data queue.

In the following example, the SUBATTR and SUBUNPT commands affect which characters are used as substitutes for attribute bytes and unprintable characters:

```
MSGID ON
RDSCREEN TERM4
OPS3581I
OPS3581I --------------  ISPF/PDF PRIMARY OPTION MENU  ---------------------
OPS3581I OPTION  ===>
OPS3581I              _
OPS3581I                                                    USERID - TSOUSER
OPS3581I  0  ISPF PARMS  - Specify terminal and user parameters    TIME - 15:35
OPS3581I  1  BROWSE      - Display source data or output listings  TERM - 3278
OPS3581I  2  EDIT        - Create or change source data           PF KEYS - 24
OPS3581I  3  UTILITIES   - Perform utility functions
OPS3581I  4  FOREGROUND  - Invoke language processors in foreground
OPS3581I  5  BATCH       - Submit job for language processing
OPS3581I  6  COMMAND     - Enter TSO command or CLIST
OPS3581I  7  DIALOG TEST - Perform dialog testing
OPS3581I  C  CHANGES     - Display summary of changes for this release
OPS3581I  T  TUTORIAL    - Display information about ISPF/PDF
OPS3581I  X  EXIT        - Terminate ISPF using log and list defaults
OPS3581I
OPS3581I
OPS3581I
OPS3581I
OPS3581I
OPS3581I
OPS3581I Enter END command to terminate ISPF.
OPS3581I
Return code:
  0 - Normal completion.
  4 - RDSCREEN command not executed. A warning message was issued.
      Virtual terminal is not defined.
      Virtual terminal is disabled.
```

This command places the screen contents into the REXX external data queue, one line per screen row. The first line in the external data queue is the operator information area line (the bottom status line on a real terminal). Screen row 1 is returned as line 2 in the queue, screen row 2 as line 3, and so on. The number of lines added to the data queue equals the number of screen rows plus one.

Due to the MSGID OFF command, these lines will not be prefaced with the OPS*nnnn* message ID.

RDSCRROW termname row

The selected row is returned as one output line.

This command runs synchronously in AOF rules and will return data in the external data queue.

```
MSGID ON
RDSCRROW TERM1 24
OPS8042I  READY
Return code:
  0 - Normal completion.
  4 - RDSCRROW command not executed. A warning message was issued.
      Virtual terminal is not defined.
      Virtual terminal is disabled.
 12 - Invalid row specified.
```

SETMODEL termname model

Use this command to set the model number to 2, 3, or 4. If the virtual terminal is enabled, this command will internally disable the virtual terminal, set the model number, and then enable the virtual terminal.

```
SETMODEL TERM1 3
Return code:
  0 - Normal completion.
  4 - virtual terminal is not defined.
```

Important! We strongly recommend you do not use this command because it will be removed in a future release.

SETTERM termname

Use this command to set the current virtual terminal. Subsequent commands with a *termname* of '*' will refer to this virtual terminal. When the REXX program begins, the current virtual terminal as used by REXX is set as the current virtual terminal undefined.

```
SETTERM TERM1
Return code:
  0 - Normal completion.
```

SETUNAME termname [user_termname]

Use this command to set the user-defined name for a virtual terminal. To remove the current user-defined name, omit the second argument.

```
SETUNAME TERM1 CICS
SETUNAME TERM1
Return code:
  0 - Normal completion.
  4 - SETUNAME command not executed. A warning message was issued.
      Virtual terminal name is not defined.
 12 - User name is already defined.
```

TRACE termname [ON|OFF]

The TRACE command is used to turn terminal tracing on or off. When terminal tracing is on, messages are written to hardcopy to display RPLs and buffers being sent and received.

```
TRACE TERM1 ON
Return code:
  0 - Normal completion.
  4 - TRACE command not executed. A warning message was issued.
      No terminals found to trace (TRACE ALL).
      Terminal name is not defined.
```

TYPE termname 3278_keyboard_text

Use this command to simulate typing at a 3278 keyboard. The operand consists of literal text and host key names. For a detailed description of HOSTKEYS and TYPE, see OPS/REXX Programming Tips in this chapter.

```
TYPE TERM1 !HOME!ERASE_EOF=x!ENTER
Return code:
  0 - Normal completion.
  4 - TYPE command not executed. A warning message was issued.
      virtual terminal is not defined.
      virtual terminal is not enabled.
 12 - Invalid keystroke (typing in protected field or
      when keyboard is inhibited for input).
```

TYPESEC termname 3278_keyboard_text

Use this command to simulate typing at a 3278 keyboard. It is exactly the same as the TYPE host command with the exception that this command does not write anything to the OPSLOG. It is provided to keep passwords from appearing in the OPSLOG.

```
TYPESEC TERM1 !HOME!ERASE_EOFsecret!ENTER
Return code:
  0 - Normal completion.
  4 - TYPESEC command not executed. A warning message was issued.
      virtual terminal is not defined.
      virtual terminal is not enabled.
 12 - Invalid keystroke (typing in protected field or
      when keyboard is inhibited for input).
```

TYPETEST termname 3278_keyboard_text

This command is provided for debugging a TYPE command. Instead of actually simulating the typing at a 3278 keyboard, the resulting keystrokes are returned as output lines, one for each key.

```
TYPETEST * !HOME!BACKTAB!ERASE_EOF'K E,D'!ENTER
 0  04  HOME
 0  0C  BACKTAB
 0  08  ERASE_EOF
 0  7D  '
 0  D2  K
 0  40
 0  C5  E
 0  6B  ,
 0  C4  D
 0  7D  '
 0  7D  ENTER
Return code:
  0 - Normal completion.
  4 - TYPE command not executed. A warning message was issued.
      virtual terminal is not defined.
      virtual terminal is not enabled.
```

In the above example, the first (one-digit) field can be 0 for a normal key, 1 for a shifted key, or 2 for a key pressed while the ALT key is held down. The next (two-digit) field is the EPI scancode (an internal code useful only for debugging purposes.) The last field is the EPI host key name.

Other EPI Host Command Descriptions

The following EPI host commands are those that do not fit into one of the above sections. Other than the HELP command, all the rest is are alphabetical order.

HELP

Use this command to display the set of recognized host commands. This command can be used when you receive a new release of the EPI to determine what the new commands are.

```
EPI HELP
```

WAIT seconds

Use this command to delay processing for a specified number of seconds (from 1 to 86400 seconds = 24 hours). Its value can be either an integer or a real number with two places available to the right of the decimal point. The time delay value is guaranteed to be at least as long as the requested value. A value of zero is valid.

Note: This command is not allowed in any AOF rules with the exception of request rules.

```
WAIT 3
Return code:
  0 - Normal completion.
```

WAITTOD hh mm ss

Use this command to hold further execution until a specific time of day has been reached. The values hh mm ss are integers, in military time, that stand for the time of day. The time must be at least one second and not more than 24 hours into the future.

Note: This command is not allowed in any AOF rules with the exception of request rules.

```
WAITTOD 12 35 0
Return code:
  0 - Normal completion.
 12 - Time specified is not 1 second to 24 hours into the future.
```

OPS/REXX Programming Tips

Use these notes in creating OPS/REXX programs for use with the EPI.

REXX Statement Transformation

A REXX statement (actually a clause in REXX terminology) recognized as a host command is transformed before it is passed to the EPI. The EPI sees only the transformed statement, not the original statement that you specified.

The following is the order in which REXX commands are processed:

- Expressions in the statement are evaluated
- The values of REXX variables are substituted for REXX variable names
- Enclosing quotes are removed from literal strings
- Multiple blanks between words are reduced to one blank

REXX Coding Considerations

REXX handles the asterisk (*) both as a wildcard character and as the multiplication character:

- Many of the host commands use an asterisk (*) to indicate an omitted parameter. If you do not place the asterisk in quotes, REXX will assume that the asterisk is the multiplication operator.

  If you enter:

  ```
  RDSCRROW  *  5
  ```

  REXX will try to multiply the variable RDSCRROW times 5. Depending on the value of RDSCRROW, REXX can fail. In the worst case, the value of RDSCRROW would be an integer, which would cause an incorrect result.

  If you enter:

  ```
  RDSCRROW  '*'  5
  ```

  REXX understands that the asterisk is used as a substitute character. When the command executes, REXX transforms the statement to:

  ```
  RDSCRROW * 5
  ```

- The REXX abuttal operator (||) accomplishes string abuttal concatenation. In the examples, the abuttal operator is only used when absolutely necessary, usually between two REXX variables. If string concatenation is performed using one or more blanks, only one blank will be passed to the EPI.
- The fact that REXX removes enclosing quotes should not be a problem. A means is provided for supplying single and double quotes in the TYPE and TYPETEST host commands. If you want to pass a single or double quote to the EPI in a host command, be sure to enclose it in another set of quotes. Only the outer quotes will be removed by REXX before passing it to the EPI.

ENQ/DEQ Notes

The ENQ/DEQ mechanism allows multiple OPS/REXX programs to share a virtual terminal. Since nothing prevents a program from getting exclusive control of a virtual terminal (ENQ) and never releasing control of it, the EPI will automatically issue a DEQ for any virtual terminals ENQd by a program under the following circumstances:

- The TCB under which the EPI ENQ was issued terminates
- The address space under which the EPI ENQ was issued terminates

In addition, the possibility exists that two REXX programs get into a deadlock situation when ENQing on multiple virtual terminals. Consider the situation of two REXX programs (PGM1 and PGM2) trying to use two virtual terminals (TERMA and TERMB) as follows:

1. PGM1 issues ENQ TERMA and gets exclusive control of TERMA.
2. PGM2 issues ENQ TERMB and gets exclusive control of TERMB.
3. PGM1 issues ENQ TERMB and waits for TERMB.
4. PGM2 issues ENQ TERMA and waits for TERMA.

Neither program will be able to continue and both virtual terminals are inaccessible by all other REXX programs. To prevent this situation, the following additional protocol is enforced by the EPI: All ENQs must be performed in alphabetical order by virtual terminal name. Therefore, in the above example, the fourth step (ENQ TERMA by PGM2) would have been rejected by the EPI, as PGM2 has already issued an ENQ for TERMB.

The TYPE Host Command

The TYPE host command is used to type on the terminal keyboard as if it were the keyboard of the device it is simulating through the currently selected virtual terminal.

The TYPE command is a combination of text and 3278 key names, as the following examples illustrate:

```
TYPE TERM1 'k e,d'!ENTER
```

Note: If there was a space between k e, d and! ENTER, the result would be the typing of an additional space. In this case, this would not be significant, but in others it might be.

The following example issues an MCS reply command, the reply number, being in the REXX variable reply_nbr, and the response in the REXX variable *mcs_response*:

```
TYPE TERM1 'r 'reply_nbr','!QUOTE1||mcs_response||!QUOTE1||!ENTER
```

Note: This example used the minimum number of abuttal operators (||).

The HOSTKEYS Host Commands

The HOSTKEYS command is not necessary under the EPI. The EPI only checks the syntax of the HOSTKEYS command.

The EPI allows the following (equivalent) syntax:

```
  TYPE TSOTERM !HOME'HELLO'!ENTER'
  TYPE TSOTERM '!HOMEHELLO!ENTER'
 'TYPE TSOTERM !HOMEHELLO!ENTER'
  TYPE TSOTERM !HOME'HELLO'!ENTER
  TYPE TSOTERM !HOMEHELLO!ENTER
```

Attribute Byte Representation in the EPI

When SUBATTR OFF is in effect, the EPI will return 3270 attribute bytes imbedded in the responses returned by the RDSCREEN and RDSCRROW commands. The attribute bytes are always in the range X'20' through X'3F' and have the following bit definitions:

```
B'001. ....'  Set for all attribute bytes
B'...1 ....'  Protected field
B'.... 1...'  Numeric field
B'...1 1...'  Autoskip field
B'.... .01.'  Low intensity  / Non-Sel-pen detectable
B'.... .01.'  Low intensity  / Sel-pen detectable
B'.... .10.'  High intensity / Sel-pen detectable
B'.... .11.'  Non-display    / Non-Sel-pen detectable
B'.... ...1'  Modified Data Tag (MDT)
```

You can use the TRANSLATE function to translate these values. For example, the following text will cause all modified fields to be preceded by an exclamation point:

```
ATTRBYTES = XRANGE(X'20', X'3F')
NEWVALUES = COPIES("!",16)
NEWSTRING = TRANSLATE(OLDSTRING, NEWVALUES, ATTRBYTES)
```

EPI Failure Recovery

The EPI distinguishes between three types of failures: system failures, session failures, and command failures, as discussed in the following sections.

EPI System Failure

When the EPI on a particular system is unable to continue, all sessions to and from that EPI will fail. Any of the following conditions can cause an EPI system failure:

- A VTAM shutdown
- Deactivation of the EPI VTAM application in VTAM
- Failure of the CA OPS/MVS address space

EPI Session Failure

When a session between the EPI and an application fails, all communication between the two ceases. Any of the following conditions can cause an EPI session failure:

- An EPI system failure will result in a failure of all sessions using that EPI.
- Deactivation of an EPI application name in VTAM will result in a failure of all sessions using that EPI application ID.
- An EPI session to a cross-domain application will fail if disruption of the communications link between two systems (lost phone connection, lost satellite link, and so on) occurs.

After an EPI session failure, no commands can be issued over that session. If the session is the only one to a particular VTAM application, all communication is lost to that application. If other sessions exist to the same application, communication may still be possible over the remaining sessions.

If possible, the EPI will write messages to the system log to identify the commands that were being processed at the time a session failed.

After a session failure, systems that have defined the session with the RETRY option will automatically attempt to reestablish the session (assuming that both systems involved in the session are still operative).

EPI Command Failures

An EPI command can fail for a variety of reasons:

- An EPI system failure
- An EPI session failure
- The application that will process the command fails

With the exception of the first two cases, the EPI automatically recovers from EPI command failures.

In most cases, the command issuer receives a message to show the specific cause of the command failure (that is, session disabled, VTAM shutdown, application logged off) and a negative return code from the last issued EPI host command.

If possible, the EPI also writes a message to the system log to identify a failing EPI command.

Security Considerations

You need to consider the points in this section with respect to the security of your EPI facility.

VTAM APPLIDs

The APPLIDs used by the EPI itself can be used to fake access to other applications, unless the APPLIDs themselves are protected. This means protecting the VTAM list libraries with RACF or CA ACF2 and using the PRTCT operand on all EPI application definitions.

Issuing Commands to Other Applications

The EPI could potentially establish a session with any VTAM application. Assuming that the correct protocols are used, nothing prevents a REXX program from accessing the VTAM application once the EPI has logged onto it.

User Exit

The user exit OPUSEX is called for every EPI command before it is executed. You can validate each request and accept or reject it in this exit.

Insert Mode

The Insert mode is not supported by the TYPE and TYPESEC commands. Neither command will reject the !INSERT keyword, but all subsequent typing is still performed in replace mode (that is, typed characters overlay existing characters on the screen).

Hardcopy Command Logging

The EPI writes a copy of every EPI command issued to the hardcopy log. If passwords were being entered during a logon sequence, they would appear in the hardcopy log. To prevent this, use the TYPESEC command instead of the TYPE command. The character string typed with a TYPESEC command is not written to the hardcopy log.

Passwords and the EPI

If you are writing dialogs that include entering passwords or other confidential information, keep in mind that even though passwords are not visible on a real screen, they are visible on a virtual screen (using the RDSCREEN or RDSCRROW or PEEK commands). Follow these guidelines:

- Secure all EPI commands through security rules.
- Store REXX programs that include passwords in read-protected libraries to prevent others from reading the password from the REXX source code.
- Do not leave a session logged on if it gives access to secured facilities, unless you can control access to the virtual terminal once it is logged on.

OMMVS—Sample OMEGAMON Interface Routine

The routine OMMVS in the REXX library (data set SYS1.OPS.CCLXEXEC documented in the *Administration Guide*) is an OPS/REXX program that issues OMEGAMON commands and gets responses returned in the external data queue of the calling program. For example, to issue the EXSY OMEGAMON command, use:

```
CALL OMMVS 'EXSY|EXECUTE'
```

The vertical bar | is the default command separator. Editing the OMMVS *separator* parameter can modify the default command separator. The EXECUTE command is the equivalent of pressing the Enter key. A DO WHILE QUEUED() loop can process the returned response lines.

You can stack multiple commands, including major and minor command combinations. For example, the following CALL command executes the LLT minor command (LinkList Table) of the XSYS major command, which returns a list of data sets in the system link list:

```
CALL OMMVS 'XSYS|LLT|EXECUTE'
```

You can make multiple calls to OMMVS and cause commands to be stacked until you call OMMVS with the EXECUTE command. The previous example could have been written:

```
CALL OMMVS 'XSYS'
CALL OMMVS 'LLT'
CALL OMMVS 'EXECUTE'
```

or

```
CALL OMMVS 'XSYS|LLT'
CALL OMMVS 'EXECUTE'
```

or

```
CALL OMMVS 'XSYS'
CALL OMMVS 'LLT|EXECUTE'
```

By default, EXECUTE presses the Enter key, which can be overridden by specifying a PF key or other attention key (PF1, ..., PF24, PA1, PA2, PA3, CLEAR, or Enter).

For example:

```
CALL OMMVS 'EXECUTE CLEAR'
CALL OMMVS 'EXSY|EXECUTE ENTER'
```

OMMVS Implementation

The following process discusses the OMMVS implementation.

- The OMMVS REXX program uses the ADDRESS EPI environment to do the following:
  - Define a virtual terminal
  - Enable it
  - Log it on to OMEGAMON/VTAM

  When a user wants to execute a sequence of commands, they are stacked in global variables (uniquely named for each address space) until an EXECUTE command is encountered.
- The OMEGAMON virtual terminal is then enqueued on exclusively and:
  - The commands are entered onto the screen.
  - Responses are scrolled through to retrieve all output, and are placed in the REXX external data queue.
  - The virtual terminal is dequeued after completion.
- The OMMVS program allows multiple users to issue commands using a single OMEGAMON virtual terminal. OMMVS takes care of the details of sharing the virtual terminal.

OMMVS Customization Variables

You must customize the OMMVS REXX program before using it. The initial section of the program contains several assignment statements for variables whose values are installation dependent.

The following variables are installation dependent:

separator

Changes the command separator from the default vertical bar (|) to a character or string of your own choice.

Note: Choosing a single character can be problematic. You must use a character that will not be interpreted as part of an OMEGAMON command.

termname

Specifies the name of the virtual terminal to be used by OMMVS. This name must be defined to VTAM, but need not be defined to the EPI.

termpswd

If the virtual terminal was defined to VTAM with a password (PRTCT= keyword on the APPL statement), then you must specify that same password here.

applname

Specifies the name of the OMEGAMON VTAM application ID.

logmode

Specifies the logmode name to be used for the virtual terminal.

subsys

If you will be using a secondary copy of CA OPS/MVS with OMMVS (a copy other than subsystem OPSS), then you must specify its four-character subsystem name here.

OM_Userid

Specifies the OMEGAMON user ID (if required by OMEGAMON).

OM_Password

Specifies the OMEGAMON user ID password (if required by OMEGAMON).

The OMEGAMON password and user ID are only required if OMVTAM security was implemented.

Disable the OMEGAMON Use of Extended Attributes

OMEGAMON can assume that your EPI virtual terminal is capable of handling extended color and attributes even though EPI terminals identify themselves as IBM 3278 type terminals that do not have extended attribute capability. This will cause the EPI to generate an error when OMEGAMON sends its first screen to an EPI virtual terminal.

To prevent this and to allow OMEGAMON to be used on an EPI virtual terminal, you must ensure that OMEGAMON disables the use of extended attributes *before* displaying the first screen.

To disable the OMEGAMON use of extended attributes

1. Ensure that one of the following OMEGAMON commands is issued before the first screen is displayed by OMEGAMON:

   ```
   .SCC DISPLAY=BASIC
   ```

   or

   ```
   .CLRX0F
   ```

   You can now use OMEGAMON on an EPI virtual terminal.

The release of OMEGAMON you are running will determine which of these two commands should be used.

CA7MVS—Sample CA 7 Interface Routine

The routine CA7MVS in the REXX library (data set SYS1.OPS.CCLXEXEC documented in the *CA OPS/MVS Administration Guide*) is an OPS/REXX program that issues CA 7 commands and gets responses returned in the external data queue of the calling program.

- This syntax issues the CA 7 LACT command:

  ```
  CALL CA7MVS 'LACT|EXECUTE'
  ```

  Vertical bar |

  The default command separator. Editing the CA7MVS *separator* parameter can modify the default command separator.

  EXECUTE command

  The equivalent of pressing the Enter key. A simple DO WHILE QUEUED() loop can process the returned response lines.

- To enter data on a multiple field screen, you can input using the split vertical bar.

  For example, This syntax places the command PF in the first (top) input field, the command I20 in the second input field, and then presses the Enter key (EXECUTE command):

  ```
  CALL CA7MVS 'PF|I20|EXECUTE'
  ```

- You can make multiple calls to CA7MVS and cause input to be stacked until you call CA7MVS with the EXECUTE command. The previous example could have been written:

  ```
  CALL CA7MVS 'PF'
  CALL CA7MVS 'I20'
  CALL CA7MVS 'EXECUTE'
  ```

  or

  ```
  CALL CA7MVS 'PF|I20'
  CALL CA7MVS 'EXECUTE'
  ```

  or

  ```
  CALL CA7MVS 'PF'
  CALL CA7MVS 'I20|EXECUTE'
  ```

By default, EXECUTE presses the Enter key, which can be overridden by specifying a PF key or other attention key (PF1, ..., PF24, PA1, PA2, PA3, CLEAR, or Enter). For example:

```
CALL CA7MVS 'EXECUTE CLEAR'
CALL CA7MVS 'LACT|EXECUTE ENTER'
```

CA7MVS Implementation

The CA7MVS REXX program uses the ADDRESS EPI environment to define a virtual terminal, enable it, and log it on to CA 7. When a user wants to execute a sequence of input commands, they are stacked in global variables (uniquely named for each address space) until an EXECUTE command is encountered.

The CA 7 virtual terminal is then enqueued on exclusively, and the input data is entered onto the screen. Responses are scrolled through to retrieve all output and are placed in the REXX external data queue. The virtual terminal is dequeued after completion.

The CA7MVS program handles scrollable output such as that from LACT, LPRRN, LJOB, and most other list commands (L*xxx*) in a special way, as follows:

- All blank lines are removed.
- Page number lines are removed.
- Duplicate column header lines are removed from page 2 and all subsequent pages.

This causes the returned output to start with a single set of column header lines, followed by detail data lines without intervening headers or blank lines, followed optionally by a command completion line from CA 7.

The intended use for the CA7MVS is primarily for command/response type applications, where a single command is used to inquire CA 7 (non-interactive commands). Commands that invoke an interactive dialog (such as QJCL) can be issued through CA7MVS, but will leave the terminal unusable for other callers of CA7MVS in that case.

The CA7MVS program does allow multiple users to issue non-interactive commands using a single CA 7 virtual terminal. CA7MVS takes care of the details of sharing the virtual terminal in such case.

CA7MVS Customization Variables

The CA7MVS REXX program must be customized before using it. The initial section of the program contains several assignment statements for variables whose values are installation dependent.

Separator

Changes the command separator from the default vertical bar (|) to a character or string of your own choice.

Note: Choosing a single character can be problematic. You must use a character that will not be interpreted as part of a CA 7 command.

termname

Specifies the name of the virtual terminal to be used by CA7MVS. This name must be defined to VTAM, but need not be defined to the EPI.

termpswd

If the virtual terminal was defined to VTAM with a password (PRTCT keyword on the APPL statement), then you must specify that same password here.

applname

Specifies the name of the CA 7 VTAM application ID.

logmode

Specifies the log mode name to be used for the virtual terminal.

subsys

If you will be using a secondary copy of CA OPS/MVS with CA7MVS (a copy other than subsystem OPSS), then you must specify its four-character subsystem name here.

CA7_Userid

Specifies the CA 7 user ID (if required by CA 7).

CA7_Password

Specifies the CA 7 user ID password (if required by CA 7).

The CA 7 password and user ID are only required if CA 7 security was implemented.

Chapter 14: Using the EPI Recording and Playback Options

This section contains the following topics:

Overview: Recording REXX EXECs

Using the EPI recording and playback options makes designing REXX automation EXECs for VTAM application sessions easier. The EPI generates a REXX EXEC containing generic routines to simplify these programming tasks:

- Finding a text string in the current session screen image
- Converting the text, if found, into REXX variables
- Fetching a screen using the GETSCRN command processor
- Pasting a sample copy of the screen image into the REXX EXEC
- Issuing commands or sending text strings to the session through SESSCMD command processors
- Monitoring the status of the current session
- Verifying that a screen image contains certain data

Requirements for Recording

Before you record screen images from an EPI session and use them to create a REXX EXEC through the EPI, you must meet these requirements:

- The session is defined to the EPI.

  Note: If defined, the session appears on the EPI Virtual Terminals List panel.

- TSO/E Version 2 is present on your system. The EPI REXX facilities require this version of TSO/E.

Note: The EPI supports Model 2 through Model 4 terminals, but we recommend that you use a Model 4 terminal for session recording. The 43 x 80 character screen size of the Model 4 terminal is large enough to display both the EPI recording command list and the full session screen image. If you use a terminal with a smaller screen, you will have to scroll the session screen more frequently.

Plan Your Recording Session

Before starting a recording session, you need plan what you want to record and how to record your choices.

To plan your recording session

1. Operate the session for a while.

   This helps you determine which tasks you want your automation EXEC to automate.

2. Select the EPI recording option and start recording, using the commands used in the EPI Recording Environment to interact with the session screen images.

   In response, the EPI inserts the captured screen data, along with comments, into the appropriate REXX routines.

3. After customizing the generic automation routines with screen data, you can use the ISPF editor to further tailor the routines and delete the automation routines you will not use.

4. You can call the ISPF editor to revise your new automation EXEC without leaving the EPI.

You have planned your session and have customized and tailored the routines.

More information:

Commands for Use in the EPI Recording Environment (see page 520)

How the Recording Option Works

Using the recording option is basically a four-task process:

1. Set up the recording environment.
2. Record the automation REXX EXEC.
3. Edit and customize the recorded REXX EXEC.
4. Test the recorded EXEC by executing it in playback mode and observing how it affects the session.

Issue Recording Commands

The EPI recording option provides a set of commands to define the recording environment, to mark text from the application screen to insert into your REXX EXEC, to enter keystrokes or to issue CA OPS/MVS command processors, and to turn recording on or off. You can issue most commands in either of two modes:

- Novice Mode

  When you issue a command in novice mode, the EPI displays a panel describing what the command does and telling you what keystrokes to enter.

- Experienced Mode

  When you issue a command in experienced mode, the EPI lets you execute that command without displaying its explanatory panel.

To issue a recording command in experienced mode

- Type the command and any other necessary keywords or characters on the command line.
- Press Enter.

  The command executes.

Important! Choose novice mode if you are new to the EPI. Experienced mode provides a shortcut for users familiar with recording commands.

Stack Recording Commands (For Advanced Users)

For advanced EPI users, the recording option offers a command stacking feature. This feature lets you issue a series of recording commands from the command line with a single Enter keystroke.

To stack commands

1. Type the commands on the command line.
2. Place a separator character such as an exclamation point (!) between the commands.

For example, the following command stack sets a variable called *varname*, appends a SAY statement using the variable to the end of the EXEC being recorded, and then shows you the revised EXEC in edit mode:

```
S varname!A SAY 'varname=' varname!E
```

Recording Environment Set Up

Before you do any recording, decide whether you want to alter the EPI recording options, which include:

- Set special control characters for use in recording sessions.
- Specify default values for SESSCMD command processors in the REXX EXECs you will record.

Changes you make to the default recording option values can take effect permanently (until you revise them again) or temporarily (for only the life of the current recording session). However, you need to make any changes before you activate session recording.

Change Recording Options Permanently

You can alter the recording options for the current recording session and have the options take effect permanently.

To permanently alter the default recording options

1. Choose the R (Record) option from the EPI Virtual Terminals List panel.

   As a result, the EPI Record and Playback Primary Menu appears:

```
-------------------- EPI Record and Playback Primary Menu ------------------
OPTION ===>                                          DATE - 04/03/11
                                                     TIME - 08:13
                                                     MODE - PROD
 0 OPTIONS   - Permanent Change to the EPI recorder options
 R RECORD    - Record EPI session REXX's
 B playBack  - Execute previously recorded EPI session EXEC
 Enter END command to terminate EPI Recording.
```

2. Type O in the Option field and press Enter.

   The following Permanent Change to the EPI Recorder Options panel appears. This panel shows you the current option settings for the recording environment:

```
----------------- Permanent Change to the EPI Recorder Options -------------
 COMMAND ===>
 Control characters:
   Relative screen location character                    (RL)..( @ )
   Absolute screen location character                    (AL)..( # )
   Command Stacking character                            (CS)..( ! )
   Cursor position character                             (CP)..( * )
 SESSCMD options:
   AUTOMATIC ENTER OPTION( NO )
   CMDWAIT ( 60  )
   MAXCMDOUT ( 200 )
   PREFIX( LINE                )
   TRUNCATE ( NO  )
Enter END command to enter option and exit, CANcel to abort
```

3. Change any default control character by typing the character you want to use over the default.

   Do not use the following as control characters:

   - A character that appears on the target session screen
   - An alphanumeric character
   - A blank

4. Change any default SESSCMD options by typing the value you want to use over the default value. The value you substitute cannot be either alphanumeric or a blank.

5. After you have finished altering the recording options:

   - Issue the END command to save your changes exit.

- Issue the CAN command to undo your changes.

Your recording options are permanently changed.

Control Characters and Defaults

The following explains the purpose of the control characters and lists the default of each:

Control Character: Relative screen location character

Marks the screen data to be searched for. This character allows you to search for data in either full-screen or split-screen mode.

Abbreviated Name: RL

Default Character: @

Control Character: Absolute screen location character

Marks the screen data to be captured and inserted into the EXEC being recorded.

Abbreviated Name: AL

Default Character: #

Control Character: Command stacking character

Separates recording commands when you issue multiple commands together from the command line of the EPI Recorder panel.

Abbreviated Name: CS

Default Character: !

Control Character: Cursor position character

Represents the current cursor position when a new session screen image appears.

Abbreviated Name: CP

Default Character: *

SESSCMD Keywords and Defaults

The following lists the keywords you can specify for SESSCMD command processors in your REXX EXEC, their purpose, and their default values:

AUTOMATIC ENTER OPTION

Determines whether the REXX EXEC automatically appends an Enter keystroke to each character string you send to a session through the SESSCMD command processor. The default value for the ENTER parameter is NO, for these reasons:

- Setting the parameter to YES gives you the option of issuing multiple SESSCMDs to place multiple character strings on the session screen-or to fill out information about multiple screens-before issuing an ENTER keystroke.
- If you issue the C, L, or N command while recording and the automatic ENTER option is set to NO, the session receives the keystroke but will not execute it until you use the K command to issue an ENTER keystroke.

Default: NO

CMDWAIT

Sets the maximum number of seconds CA OPS/MVS waits for the session to respond to a SESSCMD command processor. This number can be any value from 0 to 600.

Default: 60

MAXCMDOUT

Sets the number of lines of the SESSCMD response that should be captured and returned to the REXX EXEC. This number can be any value from 0 to 1000.

Default: 200

PREFIX

Provides the prefix for a set of REXX variables.

Default: LINE

TRUNCATE

Specifies whether CA OPS/MVS truncates the screen image display when you issue a SESSCMD command processor.

Default: NO

Change Recording Options Temporarily

You can alter the recording options for the current recording session without affecting the permanent option settings.

To change recording options temporarily

1. Issue the O (set Options) command from the EPI Recorder panel.

   In response, you will see the Temporary Change panel, which looks just like the Permanent Change to the EPI Recorder Options panel.

2. Use the same procedure to enter your temporary changes that you would use to enter permanent changes.

3. Issue the END command to save your changes or the CAN command to cancel them.

Request Temporary Changes from the Command Line

Experienced EPI users can set control characters or SESSCMD keywords from the command line of the EPI Recorder panel without displaying the Temporary Change panel.

To request temporary changes from the command line

1. Type the following text on the command line and press Enter:

   0 *abbreviation newchar*

   abbreviation is the two-character abbreviation for the control character; *newchar* is the character that will replace the default character. For example, the following text changes the absolute location character from # to $:

   ```
   0 AL $
   ```

2. Set a SESSCMD keyword, type the following text on the command line before pressing the Enter key:

   0 *keyword value*

   You must specify the complete keyword name. For example, to direct CA OPS/MVS to wait 75 seconds for a SESSCMD response, you would enter this text on the command line:

   ```
   0 CMDWAIT 75
   ```

Override the Automatic ENTER Option

You can override the automatic ENTER option by specifying either the ENTER keyword or the NOENTER keyword when you issue the C, K, L, or N commands. For example, if the automatic ENTER option is set to YES and you want to issue a tab keystroke to your session without appending an automatic ENTER keystroke, you would issue this command:

```
K TAB NOENTER
```

Choose Where to Store the REXX EXEC

After you choose a session for recording, the REXX Destination panel appears.

To choose where to store the REXX EXEC

1. On the Destination panel, enter the names of a data set and a member to store the REXX EXEC you will create in the recording session.

   Specify a cataloged data set; otherwise, you will receive an error message.

2. Press Enter.

   The destination data set for storing the REXX EXEC is created and the EPI Recorder panel displays.

Note: If you are making changes to a REXX EXEC that already exists, make sure that the EXEC points to the session you chose for the current recording session.

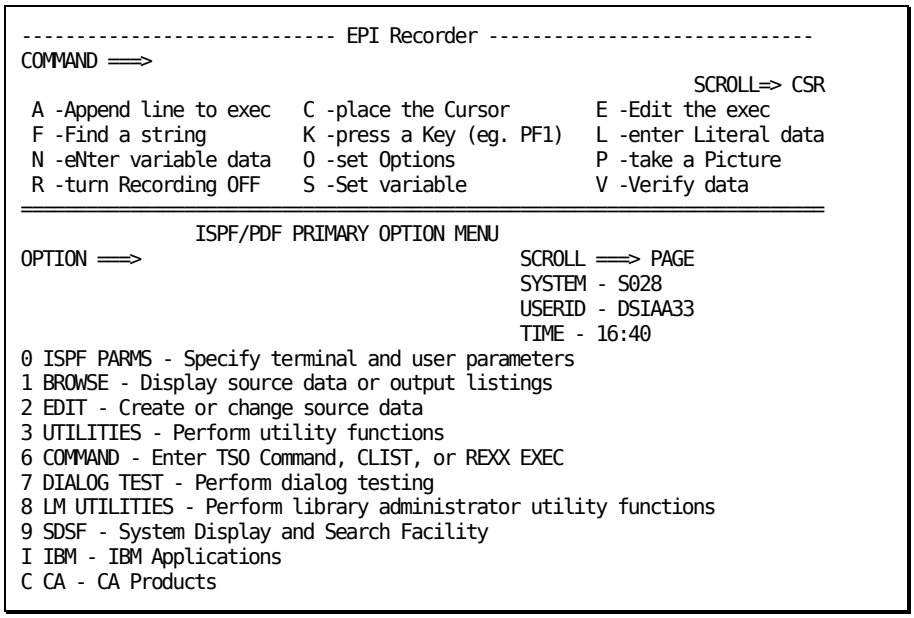
Record a Session

After you have specified a valid data set member to store your recorded REXX EXEC, the EPI displays its EPI Recorder panel, and you are ready to start recording.

To record a session

1. Review the EPI Recorder panel.

   - The bottom part of the following EPI Recorder panel displays the current screen image of the session you chose for this recording session.
   - The top (highlighted) part of the panel displays a set of commands you can use to work with text on the session screen or to make changes to the REXX EXEC you are building.



2. What you do during a recording session depends on what you want to add to or alter in the generic session automation EXEC. Typically, though, you will perform the following tasks in something close to the following order:

   a. Use the EPI V (Verify data) command to specify what screen data your session automation EXEC should look for to determine whether it is examining the right session screen.

   b. Use the EPI L (enter Literal data), C (place the Cursor), N (eNter variable data), K (press a Key) command, or all four commands to build SESSCMDs into your EXEC.

      These SESSCMDs will issue the keystrokes and commands that bring up the session display from which your EXEC pulls data.

c. Use the EPI V (Verify data) command to recheck that the correct session screen is present.

d. Issue the EPI P (take a Picture) command to insert a copy of the current session screen image into your REXX EXEC.

e. Use the EPI S (Set variable) command to insert the found screen data into a single REXX variable.

f. Use the EPI R (Turn recording off) command if or when you need to scroll the screen image to fetch data for your EXEC and then turn recording back on.

g. Use the EPI E (Edit the EXEC) command to display and revise the contents of your automation EXEC through the ISPF editor.

h. Use the EPI Playback option to test and debug your session automation EXEC.

You have recorded your session.

How the Recording Process Works

During recording, you customize the EPI generic automation REXX EXEC, importing data from session screens, and building SESSCMD command processors to either issue commands to the session or respond to prompts from the session.

Which data you import and which SESSCMD command processors you place in your REXX EXEC depend on the type of session management task you want the EXEC to automate.

Example: Recording Process

Suppose that you want the EXEC to automate extracting statistics about terminal response time from CA NetSpy screens. You would do the following:

1. First you need to insert into your EXEC SESSCMDs that cause CA NetSpy to display its Terminal Response Time Analysis screen.
2. Then, you would need to capture some of the information that this CA NetSpy screen displays-such as the average and last response time for transactions on the terminal-into REXX variables.

Commands for Use in the EPI Recording Environment

The following briefly explains what you can do with each command provided on the EPI Recorder panel, including the commands mentioned above. From the EPI Recorder panel, you can access online information about these commands. To do so, place the cursor on the command name and press the PF1 key.

A (Append line to EXEC)

Appends a text string you specify to the last recorded line in your REXX EXEC.

C (place the Cursor)

Places the cursor at a location (that you specify) on the session screen image.

E (Edit the EXEC)

Allows you to edit the REXX EXEC being recorded.

F (Find a string)

Finds a character string you specify on the session screen image.

K (press a Key)

Lets you issue an aid key or escape instruction to the session.

L (enter Literal data)

Inserts a SESSCMD command processor containing the literal string you specify into your REXX EXEC.

N (eNter variable data)

Inserts a SESSCMD command processor containing the REXX variables you specify into your REXX EXEC.

O (set Options)

Allows you to temporarily or permanently reset the recording environment for the current recording session.

P (take a Picture)

Captures the current session screen image and places it in your REXX EXEC as a comment.

R (turn Recording ON/OFF)

Starts or stops session recording.

S (Set variable)

Lets you set a variable.

V (Verify data)

Allows you to verify that you are on the proper screen image.

Marking Text to Find on or Fetch from a Screen

The next few sections show examples of how to mark the text you want your session automation EXEC to import from or find on a session screen.

Place the Cursor on a Screen Field

When placing the cursor on a screen field, you can specify the following:

- An absolute position if your target is screen text that never changes
- The cursor position relative to other screen data if the cursor will go on screen text that can vary

To have your EXEC place the cursor on a specific piece of screen text

1. Issue the C command at the command line.

   The following novice mode screen for the C command appears. This illustration shows a CA NetSpy statistics screen displayed for session recording. Suppose that you want to place the cursor at the CA NetSpy command line (shown at the bottom of the screen).

```
------------------------------ PLACE THE CURSOR ------------------------------
 COMMAND ===>                                                 SCROLL ===> CSR
     To specify Relative Positioning, use @ to mark the screen data
     to search for.  Then use # to mark where you want the script to place
     the cursor after the data is found.  To specify Absolute Positioning,
     use # to mark the position where the cursor will be placed.
     The default for SESSCMD is NOENTER
   ---------------------------------------------------------------------------
  HOST=S032              MEASURED INTERVAL=  3.8M  27MAR95.178 FRI 15:18:49 ***
                      *** NETSPY RESPONSE TIME STATISTICS (INTERVAL) ***
  APPL      AVGE=RSP    WORST    NO.    NO.     NO.    NO.  INPUT OUT-SZE OUT-SZE
  NAME     HOST   NET    HOST   SESS  INPUT  OUTPUT NETRSP   SIZE TRANSCT  WRITE
  ----------------------------  NETSPY= S032  ---------------------------------
  NETSPY32 0.1    0.0     0.6       1    8      8      8      6     1629    1629
  TS032    0.6    0.0     1.7       8    8      9      8      7     1366    1214
  ----------------------------  NETSPY= S028  ---------------------------------
  NETSPY28 0.0    0.0     0.4       4   16     16     16      8     1246    1246
  TS028    1.1    0.1     8.8      51   42     68     38     12      470     290
  CICSA    0.4    0.1     0.9       3  210    226    186    136     1575    1108
  CICST    0.3    0.2     0.6       2   22     24     20    130     1525    1075
  PRESS PFK1= HELP, 3= MENU, 5= DA, 6= DT, 9=DNCPS, 10= DLINES, 12=LOGOFF
   OR ENTER A NEW COMMAND BELOW
--->
```

2. Type one or more control characters over screen text to indicate where to place the cursor.

   You can specify an absolute position for the cursor if your target is screen text that never changes. Or, you can indicate the cursor position relative to other screen data if the cursor will go on screen text that can vary.

3. Assuming that the absolute screen location character is # and the relative screen location character is @:

   - Type C on the EPI Recorder panel command line at the top of the screen
   - Type # to the right of the arrow on the CA NetSpy command line
   - Then type @ over the first character and the last character in the word BELOW as follows:

```
OR ENTER A NEW COMMAND @ELO@
---> #
```

The EPI first finds the screen text overtyped with @ characters, shown in Find a Text String on a Screen (see page 523), and then places the cursor on the CA NetSpy command line.

Find a Text String on a Screen

You can mark a text string that you want your automation EXEC to find on a session screen.

To find a text string on a screen

1. Type the absolute screen location character over the first and the last character in the string.

   This marks the text string that your automation EXEC should find on a session screen.

2. Issue the F (Find a string) command from the command line.

If the text to be found is a changeable value (like the host response time values in the CA NetSpy Statistics Screen), your automation EXEC will have to find that value in relation to fixed screen text, such as a field name.

Example: Find a text string on a screen

Suppose that you want to find the value of the HOST=*systemid* field shown in the CA NetSpy Statistics Screen. Because this system ID will vary, you need to use relative positioning to indicate where the value is on the screen. On this screen, the HOST part of the field is always constant no matter which system ID appears. Therefore, you can direct your REXX EXEC to find the location of the system ID value as it relates to the text string HOST. To do so, you would type over the HOST=*systemid* field as shown in the following example:

Original screen text...	As it looks after overtyping...
HOST=S032	@OST@#03#

This causes your automation EXEC to find the system ID, S032, indicated by the # characters.

REXX Variables That the F Command Sets

When you find a string using the F command, the EPI sets values for the following REXX variables in the EXEC you are recording:

FINDSTR_RP

Stores the relative location string

FINDSTR_RP_ROW

Stores the screen row where the relative location string appears

FINDSTR_RP_COL

Stores the screen column where the relative location string appears

FINDSTR_RP_LENGTH

Stores the number of characters in the relative location string

FINDSTR_RP_OFFSET

Stores the offset position in a single integer

FINDSTR_RP_RC

Stores the return code from the Find operation

FINDSTR_AP

Stores the absolute location string

FINDSTR_AP_ROW

Stores the screen row where the absolute location string appears

FINDSTR_AP_COL

Stores the screen column where the absolute location string appears

FINDSTR_AP_LENGTH

Stores the number of characters in the absolute location string

FINDSTR_AP_OFFSET

Stores the offset position in a single integer

FINDSTR_AP_RC

Stores the return code from the Find operation

Return Codes

Possible return code values for FINDSTR_RP_RC and FINDSTR_AP_RC are:

0

The EPI found the string.

4

The EPI did not look for the relative location or absolute location string.

8

The EPI did not find the string.

Mark Screen Text to Assign to a REXX Variable

The S (Set variable) command of the EPI allows you to fetch a piece of screen text and assign its value to a single REXX variable. The experienced mode version of the S command has this format:

```
S varname
```

You can use either absolute or relative positioning to specify which screen text to capture depending on whether the text you are capturing can vary. For example, if you want to capture the text 3.8M from the CA NetSpy Statistics Screen and make that text the value of a variable called MEASURED_INTERVAL, you would type over the appropriate screen text as follows:

```
MEASURED @NTERVAL@ #3.8M#
```

Then, you would issue this command:

```
S MEASURED_INTERVAL
```

As a result, the EPI finds the screen text marked with the @ character, inserts the MEASURED_INTERVAL variable into your automation EXEC, and gives it the value 3.8M.

Note: You use the absolute screen location character to mark the start and end of the screen text to be captured.

Insert Literal Strings or Variables into SESSCMDs

Using the EPI C, K, L, and N commands, you can build SESSCMDs into your REXX EXEC to issue keystroke and command instructions or text strings to automate session management. The L and N commands not only specify the text that a SESSCMD sends to the session, they also allow you to specify which input field or fields will receive the sent text.

The next few paragraphs explain how to direct a literal string or a variable to a specific input field or fields on the screen. Online help for the C, K, L, and N commands also provides usage information.

Direct a Literal String to an Input Field

To specify where a literal string sent by a SESSCMD will go on your session screen, do either of the following:

Absolute Positioning Method

1. On the EPI Recorder panel, type an L on the command line and press Enter.
2. Type the literal string into the appropriate input field or fields on the session screen and press Enter.

Relative Positioning Method

1. On the EPI Recorder panel, type an L on the command line and press Enter.
2. Type the relative screen location character over the first and the last character in the screen data (for example, the input field name) that the SESSCMD command processor should look for before sending the literal string.
3. Type the literal string into the appropriate input field on the session screen and press Enter.

For example, to enter the command text HELP on the CA NetSpy Statistics Screen, you would type over the command line characters as shown in the following example:

Original screen text...	As it looks after overtyping, with literal text HELP added...
--->	@--@ HELP

Direct a Variable to an Input Field

The N (enter variable data) command allows a SESSCMD command processor to send any of the following data to a session:

- REXX variable or variables and literal text, sent to multiple input fields on the screen
- REXX variable or variables and literal text, sent to a single input field on the screen
- A single REXX variable, sent to multiple input fields
- Multiple REXX variables, sent to a single input field

To specify which input fields will receive this data

1. On the EPI Recorder panel, type an N on the command line. The novice mode display for the N command will appear.
2. Type the relative screen location character over the first and the last character in the screen data (for example, the input field name) that the SESSCMD command processor should look for before sending the REXX variable, literal data, or both.
3. Type the absolute screen location character into the input field or fields on the session screen that will receive input from the SESSCMD. Press Enter.

For example, suppose that you have a REXX variable called HELP_CMD set to the value HELP. To have your SESSCMD place this variable on the command line of the CA NetSpy Statistics Screen, you would type over the command line characters as shown in the following example:

Original screen text...	As it looks after overtyping, with # representing the word HELP...
--->	@--@ #

Edit Your Customized Automation EXEC

The EPI allows you to edit the automation REXX EXEC you are building without ending your recording session.

To edit your customized automation EXEC

1. Issue the E command from the EPI Recorder panel.

   The EPI displays the text of your EXEC using the ISPF editor.
2. Revise the EXEC using the ISPF editing commands.
3. When you have finished editing, issue the END command from the command line of the ISPF edit panel.

   In response, the EPI returns you to the EPI Recorder panel.

Test-run Your EXEC with the Playback Option

The playback option allows you to test-run an automation REXX EXEC in the EPI environment, enabling you to revise the EXEC easily if it does not affect session activity as you intended. The playback option tests only EXECs created using the recording option of the EPI.

To playback a recorded REXX EXEC

1. Choose option B (playBack) from the EPI Record and Playback Primary Menu.

   As a result, you see a panel asking you to enter the name of the data set and member where the EXEC to play back resides.

2. Enter the data set name and member name.

   Playback begins.

How Playback Works

After you supply the member and data set information for the EXEC to play back, the EPI follows the following process:

1. The playback function:
   - Starts running the selected REXX EXEC
   - Displays the current screen image for the affected session

   This results in short messages describing the first keystroke or keystrokes the EXEC will issue.
2. Press enter.

   The first SESSCMD command processor in the EXEC executes.
3. The playback function does the following:
   - Displays new session screen image that reflects the response of the session to the SESSCMD Displays the current screen image for the affected session
   - Freezes the screen image

   This provides time for you to study the frozen screen image and observe how the SESSCMD affected the session.
4. Press Enter

   The playback function displays the next screen image.
5. The playback function displays new messages describing what action the next SESSCMD in the EXEC will do.

   The next SESSCMD executes.
6. The playback function freezes the screen image again.

   This lets you review the effect of the SESSCMD that just executed.
7. Press Enter again.

   The next session screen image appears and steps 5 through 7 repeat until the EXEC finishes executing.
8. The playback function:
   - Returns you to the EPI Recording Primary Menu, if your EXEC contains no SAY statements to display output from the EXEC
   - Uses the SAY statements in the EXEC to write output data to your screen

Record an EXEC to Automate Info/Management Inquiries

The following sample scenario uses the EPI to create a REXX EXEC that automates fetching status information about customer support incidents from a CA archive system based on the IBM Info/Management product. You first set up an EPI recording session with the archive system. Then, you use the EPI recording commands to build an EXEC including subroutines that:

- Issue the keystrokes required to display (and exit from) the appropriate CA and Info/Management screens
- Ensure that the EXEC displays the correct screens
- Respond to screen prompts
- Pull the desired incident status information into a REXX variable

The sample EXEC contains both the CA generic automation routines and a driver section containing REXX code generated by commands issued during recording. The next few sections describe only the code in the driver section.

To create the sample automated inquiries EXEC

1. Select the R (Record) option from the EPI Virtual Terminals List panel.
2. Choose the R (Record) option from the EPI Record and Playback Primary Menu.
3. Choose a data set member to store the recorded EXEC.

   The EPI Recorder panel appears, displaying the TSO session screen image and a list of recording commands.

More information:

Choose Where to Store the REXX EXEC (see page 517)

The Opening Text of the Driver Section for a Recorded EXEC

The driver section of this sample REXX EXEC contains the user-defined REXX code that drives EXEC execution, including subroutines created while recording the session.

The following illustration shows you the user-modifiable text of the first part of the driver section:

```
/* If you need to PARSE input parameters, use a statement similar to */
/* the following.                                                     */
/*                                                                    */
/* PARSE VAR parm parm_x "," parm_y "," parm_z .                      */
/*                                                                    */
/* The variable "parm" contains all input parms except "DEBUG" and    */
/* "PLAYBACK", which were stripped out in the unmodifiable            */
/* initialization section of this EXEC.                               */
/*--------------------------------------------------------------------*/
/* ----- You must include the following PARSE statement  --------     */
 PARSE VAR parm inc_#        /* inc_# is the name of a REXX variable */
/*--------------------------------------------------------------------*/
/* The following SESSCMD keywords are set via the O command on the EPI */
/* Recorder panel. STOPMSG is the only keyword not set from the panel. */
/* If using STOPMSG, then set the following STOPMSG variable:       */
/* STOPMSG = "STOPMSG(string1,string2,string3)"                      */
/*--------------------------------------------------------------------*/
 CMDWAIT   = ''
 MAXCMDOUT = ''
 PREFIX    = ''
 TRUNCATE  = ''
 STOPMSG   = ''
/*--------------------------------------------------------------------*/
/* Call external function ATMOPOOL and set REXX variable epi_session */
/* The function will bind the session and return the CA OPS/MVS      */
/* session ID to use during the REXX procedure.                      */
/*--------------------------------------------------------------------*/
pool_returned_data = ATMOPOOL("POOL=INFOMAN" "HOME=YES")
PARSE UPPER VAR pool_returned_data epi_session home_rc
IF epi_session = '' THEN DO
  rc = "No sessions available for BIND"
  SIGNAL ERROR_HANDLER
END
IF home_rc > 4 THEN DO
  rc = "HOME_RC ="home_rc
  SIGNAL ERROR_HANDLER
END
/*--------------------------------------------------------------------*/
/* Call ATM_GETSCRN to fetch a screen.                                */
/*--------------------------------------------------------------------*/
CALL ATM_GETSCRN
```

```
/*-------------------------------------------------------------------*/
/* If PLAYBACK mode active, display first screen.                    */
/*-------------------------------------------------------------------*/
IF playback = "ON" THEN
  CALL PLAYBACK
```

Build the Rest of the Driver Section Using Recording Commands

To create the rest of the drive section of the sample EXEC, you would issue the following recording commands in novice mode as follows:

To build the rest of the driver section using recording commands

1. Type this text:

   - Issue the V (Verify data) command on the command line of the EPI Recorder panel.
   - On the TSO session display, block the TSO READY prompt by typing the relative position character over the first and last characters, as follows:

     ```
     @EAD@
     ```

   To produce these results:

   Enable your REXX EXEC to locate the READY prompt and verify that a TSO session is displayed. Issuing this V command builds the following subroutine into your REXX EXEC:

   ```
   CALL ATM_VERIFY "VERIFY_RP(READY)"
   ```

2. Type this text:

   - On the command line of the EPI Recorder panel, issue the L (enter Literal data) command.
   - On the TSO session display, block the READY prompt using the relative position character.
   - Type ISPF seven bytes to the right of @EAD@ on the TSO screen.

   Using a SESSCMD command processor, the EPI sends the text ISPF to the session and enters to the right of the READY prompt. If the automatic ENTER option is set to YES, an ENTER keystroke will also be sent.

   To produce these results:

   Invokes the ISPF Primary Option Menu. Issuing the L command builds the following subroutine into your EXEC:

```
CALL ATM_PLACE_CURSOR,
"INPUT(ISPF                    "||,
"                              "||,
"                               )",
"PLACE_CURS_STR(READY)",
"PLACE_CURSOR_RP(7)",
"ENTER(YES)"
```

3. Type this text:

   - On the command line of the EPI Recorder panel, issue the V (Verify data) command.
   - On the ISPF main menu, block the phrase, ISPF Primary Option Menu using the relative position character as follows:

     ```
     @SPF/PDF PRIMARY OPTION MEN@
     ```

   To produce these results:

   Enable your REXX EXEC to determine whether the main ISPF menu is displayed. Issuing this V command builds the following subroutine into your EXEC:

```
CALL ATM_VERIFY
"VERIFY_RP(ISPF/PDF PRIMARY OPTION    MENU)"
```

4. Type this text:

   - On the command line of the EPI Recorder panel, issue the L command.
   - Block the screen text OPTION ===> using the relative position character as follows:

     ```
     @PTION ===@
     ```

   - Type PROD on the command line of the ISPF menu, seven bytes to the right of the O in OPTION.

   Using a SESSCMD command processor, the EPI sends the text PROD to the session and enters it to the right of the command line arrow. If the automatic ENTER option is set to YES, an ENTER keystroke will also be sent.

   To produce these results:

   Display the CA Production Applications menu and place the cursor on the command line of the menu. Issuing this L command builds the following subroutine into your EXEC:

   ```
   CALL ATM_PLACE_CURSOR,
    "INPUT(PROD                    )",
    "PLACE_CURS_STR(OPTION ===>)",
    "PLACE_CURSOR_RP(12)",
    "ENTER(YES)""
   ```

5. Type this text:

   - On the command line of the EPI Recorder panel, issue the V command.
   - On the production applications menu, block the phrase, CA Production Applications, using the relative location character as follows:

     ```
     @CA PRODUCTION APPLICATION@
     ```

   To produce these results:

   Enable your REXX EXEC to determine whether the correct panel is displayed. Issuing this V command builds the following subroutine into your EXEC:

   ```
   CALL ATM_VERIFY "VERIFY_RP(CA PRODUCTION APPLICATIONS)"
   ```

6. Type this text:

   - On the command line of the EPI Recorder panel, issue the L command.
   - On the ISPF menu, block the screen text OPTION ===> using the relative location character as follows:

     ```
     @PTION ===@
     ```

   - Type LS on the command line of the ISPF menu.

   Using a SESSCMD command processor, the EPI sends the text LS to the session and enters it to the right of the Option field. If the automatic ENTER option is set to YES, an ENTER keystroke will also be sent.

   To produce these results:

   Display the logon panel for the CA Support System and place the cursor on the command line. Issuing this L command builds the following subroutine into your EXEC:

   ```
   CALL ATM_PLACE_CURSOR,
   "INPUT(LS                          )",
   "PLACE_CURS_STR(OPTION ===>)",
   "PLACE_CURSOR_RP(12)",
   "ENTER(YES)"
   ```

7. Type this text:

   - On the command line of the EPI Recorder panel, issue the V command.
   - On the logon panel, type the relative location character over the first and last characters of the text CCL09NEW.

   To produce these results:

   Enable your REXX EXEC to confirm that the logon panel is displayed. Issuing this V command builds the following subroutine into your EXEC:

   ```
   CALL ATM_VERIFY "VERIFY_RP(CCL09NEW)"
   ```

8. Type this text:

   - On the command line of the EPI Recorder panel, issue the L command.
   - On the logon panel, block the screen text ===> using the relative location character as follows:

     ```
     @==@
     ```

   Using a SESSCMD command processor, the EPI sends the text 2 to the session and enters it to the right of the ===> field. If the automatic ENTER option is set to YES, an ENTER keystroke will also be sent.

   To produce these results:

   Display the Info/Management Inquiry panel and place the cursor on the command line. Issuing this L command builds the following subroutine into your EXEC:

   ```
   CALL ATM_PLACE_CURSOR,
   "INPUT(2                    "||,
   "                            )",
   "PLACE_CURS_STR(===>)",
    "PLACE_CURSOR_RP(5)",
    "ENTER(YES)"
   ```

9. Type this text:

   - On the command line of the EPI Recorder panel, issue the V command.
   - On the inquiry panel, block the screen text CCL1IC0E using the relative location character as follows:

     ```
     @CL1IC0@
     ```

   Using a SESSCMD command processor, the EPI sends the text 2 to the session and enters it to the right of the ===> field. If the automatic ENTER option is set to YES, an ENTER keystroke will also be sent.

   To produce these results:

   Enable your REXX EXEC to confirm that the inquiry panel is displayed. Issuing this V command builds the following subroutine into your EXEC:

   ```
   CALL ATM_VERIFY "VERIFY_RP(CCL1IC0E)"
   ```

10. Type this text:

    - On the command line of the EPI Recorder panel, issue the L command.
    - On the logon panel, block the screen text ===> using the relative location character as follows:

      ```
      @==@
      ```

    - Type 1 on the command line of the inquiry panel.

    Using a SESSCMD command processor, the EPI sends the text 1 to the session and enters it to the right of the ===> field. If the automatic ENTER option is set to YES, an ENTER keystroke will also be sent.

    To produce these results:

    Display the Incident Search panel and place the cursor on the command line. Issuing this L command builds the following subroutine into your EXEC:

    ```
    CALL ATM_PLACE_CURSOR,
     "INPUT(1                    "||,
     "                            )",
     "PLACE_CURS_STR(===>)",
     "PLACE_CURSOR_RP(5)",
     "ENTER(YES)"
    ```

11. Type this text:

    - On the command line of the EPI Recorder panel, issue the V command.
    - On the Incident Search panel, type the relative location character over the first and last characters of the text CCL2SI0X and the absolute location character over the first and last characters of INCIDENT SEARCH, as follows:

      ```
      @CL2SI0@ #NCIDENT SEARC#
      ```

    To produce these results:

    Enable your REXX EXEC to confirm that the search panel is displayed. Issuing this V command builds the following subroutine into your EXEC:

    ```
    CALL ATM_VERIFY,
    "VERIFY_RP(CCL2SI0X)",
     "VERIFY_AP(INCIDENT SEARCH)",
     "REL_POS(30)",
     "LEN_AP(15)"
    ```

12. Type this text:

    - On the command line of the EPI Recorder panel, issue this command:

      ```
      L NOENTER
      ```

    - On the command line of the Incident Search panel, type this text:

      ```
      90
      ```

    Using a SESSCMD command processor, the EPI sends the text 90 to the session and enters it to the right of the ===> field. The NOENTER clause on the L command lets you override the automatic ENTER option setting of YES, allowing you to set the REXX variable described below before displaying the Table Display Options panel.

    To produce these results:

    Send Info/Management code 90 to the session with no ENTER keystroke. Issuing this L command builds the following subroutine into your EXEC:

    ```
    CALL ATM_PLACE_CURSOR,
     "INPUT(90                  "||,
     "                          )",
     "PLACE_CURS_STR(===>)",
     "PLACE_CURSOR_RP(5)",
     "ENTER(NO)"
    ```

13. Type this text:

    - On the command line of the EPI Recorder panel, issue this command:

      ```
      N INC_#
      ```

    - On the Incident Search panel, type the absolute location character (#) over the first character of the value in the Incident # field.

    To produce these results:

    Fetch the incident number in the Incident # field and place its value in a REXX variable, INC_#. Issuing this N command builds the following subroutine into your EXEC:

    ```
    CALL ATM_SESSCMD,
     "INPUT("INC_#")",
     "ROW(5)",
     "COLUMN(29)",
     "ENTER(YES)"
    ```

14. Type this text:

    - On the command line of the EPI Recorder panel, issue the V command.
    - On the Table Display Options panel, type the relative location character over the first and last characters of the text CCL3II5I and the absolute location character over the first and last characters of TABLE DISPLAY OPTIONS, as follows:

      ```
      @CL3II5@ #ABLE DISPLAY OPTION#
      ```

    To produce these results:

    Enable your REXX EXEC to confirm that the Table Display Options panel is displayed. Issuing this V command builds the following subroutine into your EXEC:

    ```
    CALL ATM_VERIFY,
     "VERIFY_RP(CCL2SI0X)",
     "VERIFY_AP(INCIDENT SEARCH)",
     "REL_POS(20)",
     "LEN_AP(21)"
    ```

15. Type this text:

    - On the command line of the EPI Recorder panel, issue the L command.
    - On the Table Display Options panel, block the screen text CCL3II5I using the relative location character as follows:

      ```
      @CL3II5@
      ```

    - Type 2 on the command line of the Table Display Options panel.

    Using a SESSCMD command processor, the EPI sends the text 2 to the session and enters it to the right of the ===> field.

    To produce these results:

    Selects display option 2, which displays an Info/Management panel containing information about specific support incidents. Issuing this L command builds the following subroutine into your EXEC:

    ```
    CALL ATM_PLACE_CURSOR,
     "INPUT(2                  "||,
     "                          )",
     "PLACE_CURS_STR(===>)",
     "PLACE_CURSOR_RP(5)",
     "ENTER(YES)"
    ```

16. Type this text:

    - On the command line of the EPI Recorder panel, type this command.

      ```
      S REC_#
      ```

    - On the Info/Management panel, block the screen text “RECORD #” using the relative location character as follows::

      ```
      @RECORD @
      ```

    To produce these results:

    Find the value in the Record # field and assign its value to a REXX variable called REC_#. Issuing this S command builds the following subroutine into your EXEC:

    ```
    REC_# = ATM_SET_VAR(,
    "SET_VAR_RP(RECORD #)",
    "START_AP(80),
    ""LEN_AP(8)")
    ```

    The EPI also inserts the following subroutine, which compares the value of the REC_# variable to the value entered earlier for the INC_# variable and issues an error message if the values do not match:

    ```
    IF REC_# <> INC_# THEN DO
    SAY"RECORD NUMBER" REC_# "NOT FOUND"
    SIGNAL EXIT_RC8
    END
    ```

17. Type this text:

    - On the command line of the EPI Recorder panel, type this command:

      ```
      S STATUS
      ```

    - On the incident information panel, block the screen text ST using the relative location character as follows:

      ```
      @@
      ```

    To produce these results:

    Find the value in the Incident Status field and assign its value to a REXX variable called STATUS. Issuing this S command builds the following subroutine into your EXEC:

    ```
    STATUS = ATM_SET_VAR(,
    "SET_VAR_RP(ST)",
    "START_AP(80),
    ""LEN_AP(1)")
    ```

18. Type this text:

    - On the command line of the EPI Recorder panel, type this command:

      ```
      A
      ```

    To produce these results:

    The EPI displays the last group of lines in your EXEC in edit mode. You then insert the following text at the end of the EXEC:

    ```
    SAY "STATUS FOR INCIDENT" INC_# "IS" STATUS
    ```

19. Type this text:

    - On the command line of the EPI Recorder panel, type this command:

      ```
      K @3
      ```

    To produce these results:

    Issue a PF3 keystroke to return you to the Table Display Options panel. Issuing this K command builds the following subroutine into your EXEC:

    ```
    CALL ATM_SESSCMD "INPUT(@3)     ENTER(YES)"
    ```

20. Type this text:

    - On the command line of the EPI Recorder panel, issue the L command.
    - On the Table Display Options panel, block the screen text CCL3II5I using the relative location character as follows:

      ```
      @CL3II5@
      ```

    - Type BACK on the command line of the Table Display Options panel.

    Using a SESSCMD command processor, the EPI sends the text BACK to the session and enters it to the right of the ===> field.

    To produce these results:

    Ensure that you are back at the Table Display Options panel, and then issue the BACK command to return to the Incident Search panel. Issuing this L command builds the following subroutine into your EXEC:

    ```
    CALL ATM_PLACE_CURSOR,
    "INPUT(BACK                  "||,
    "                          )",
    "PLACE_CURS_STR(CCL3II5I)",
    "PLACE_CURSOR_RP-155)",
    "ENTER(YES)"
    ```

21. Type this text:

    - On the command line of the EPI Recorder panel, issue the L command.
    - On the Incident Search panel, block the screen text CCL2SI0X using the relative location character as follows:

      ```
      @CL2SI0@
      ```

    - Type 60 on the command line of the Table Display Options panel.

    Using a SESSCMD command processor, the EPI sends the text 60 to the session and enters it to the right of the ===> field.

    To produce these results:

    Ensure that you are back at the Incident Search panel, and then issue the BACK command to return to the inquiry panel. Issuing this L command builds the following subroutine into your EXEC:

    ```
    CALL ATM_PLACE_CURSOR,
    "INPUT(60                  "||,
    "                          )",
    "PLACE_CURS_STR(CCL2SI0X)",
    "PLACE_CURSOR_RP-155)",
    "ENTER(YES)"
    ```

22. Type this text:

    - On the command line of the EPI Recorder panel, type the L command.
    - On the inquiry panel, type the absolute location character over column 7 of the first screen row.
    - Type Q on the command line of the Table Display Options panel.

    Using a SESSCMD command processor, the EPI sends the text Q to the session and enters it to the right of the ===> field.

    To produce these results:

    Enter a Q command to exit from the CA Support System. Issuing the L command of the EPI Recorder inserts the following subroutine into your EXEC:

    ```
    CALL ATM_SESSCMD,
     "INPUT(Q                  "||,
     "                          )",
     "ROW(1)",
     "COLUMN(7)",
     "ENTER(YES)"
    ```

23. Type this text:

    On the command line of the EPI Recorder panel, type this command:

    ```
    K @3
    ```

    To produce these results:

    Issue a PF3 keystroke to return to the CA Production Applications menu. Issuing this K command builds the following subroutine into your EXEC:

    ```
    CALL ATM_SESSCMD "INPUT(@3)  ENTER(YES)"
    ```

24. Type this text:

    On the command line of the EPI Recorder panel, type this command:

    ```
    K @3
    ```

    To produce these results:

    Issue a PF3 keystroke to return to the ISPF main menu. Issuing this K command builds the following subroutine into your EXEC:

    ```
    CALL ATM_SESSCMD "INPUT(@3) ENTER(YES)"
    ```

Chapter 15: Enhanced Console Facility

This section contains the following topics:

Overview

The Enhanced Console Facility(ECF) component of CA OPS/MVS enhances the capabilities of the z/OS existing console support by providing the following functions to operators using MCS consoles:

- An operator can run a single TSO command or a command procedure written in either OPS/REXX or the TSO CLIST language. This function is implemented using the CA OPS/MVS Server Facility, so you will see the term OSF associated with it. The command procedure can use the OPSCMD command supplied with CA OPS/MVS to issue z/OS, JES, and subsystem commands. If z/OS runs under VM, then VM commands can be issued. If the IMS Operation Facility (IOF) optional feature is installed, then IMS commands can also be issued.

  The interactive facilities of TSO do not have to be functioning for CA OPS/MVS to run TSO commands and CLISTs. CA OPS/MVS uses the TSO service routines, which are available to any address space, to execute commands issued from z/OS consoles through the ECF in one of the CA OPS/MVS Server address spaces.

- An operator can log on to the ECF, which creates a line-mode TSO session. The most common use of such a session is to run TSO EDIT to repair a member of SYS1.PROCLIB (for instance, the JES procedure) that is required to bring TSO up. Such a use of the ECF is called a system rescue.

  To use the ECF for system rescue, you must be able to run CA OPS/MVS with SUB=MSTR. SUB=MSTR allows CA OPS/MVS to be started independently of JES, which is necessary because JES might need rescuing.

Note: For more information, see the *CA OPS/MVS Administration Guide*.

Concepts

The ECF component is an integral part of the CA OPS/MVS base product, and is available as soon as the CA OPS/MVS main product address space finishes initialization. As CA OPS/MVS is usually installed, this is **before** JES, VTAM, and TSO are started. The ECF monitors commands entered from all z/OS consoles looking for one that begins with one of its special characters, as discussed in the following sections: Commands Processed by the OSF and Commands Processed by the ECF in this chapter.

Note: Enter ECF commands from real consoles only.

Commands Processed by the OSF

When the ECF sees a command that begins with an exclamation mark (!), it routes the command to one of its OPSOSF (Server) address spaces for processing.

An exclamation mark is the default OSF character string. It can be changed to any other 1- to 3-character string (other than the current ECF command character string) with the OPSPRM OPS/REXX function or the OPSPARM command. For additional information, see the OSFCHAR parameter in the *Parameter* Reference.

What follows that character must be one of the following:

- The name of a TSO command (that is, LISTC) followed by its operands, if any.
- A percent sign (%) followed by the name of a command procedure that has been written in the TSO CLIST language and has been put in one of the libraries in the //SYSPROC concatenation specified in the JCL of the OPSOSF procedure. Such procedures can contain any combination of TSO commands and CLIST logic, but they will almost always contain calls to the CA OPS/MVS OPSCMD command to issue system commands and retrieve the responses they generate.

  A CLIST that has not been stored in //SYSPROC can still be run by calling it directly through the TSO EXEC command.
- OI followed by the name of a command procedure that has been written in OPS/REXX and has been placed in one of the libraries in the //SYSEXEC concatenation specified in the JCL of the OPSOSF procedure. Again, while in theory any OPS/REXX program can be called, the assumption is that these programs perform some operational task by their routing commands to ADDRESS OPER.

  OI is an alias of OPSIMEX. OPSIMEX could be used in place of OI, but it is considerably less convenient to type.
- OX followed by the name of a data set containing a command procedure written in OPS/REXX. OX is an alias of OPSEXEC, and must be used if the OPS/REXX command procedure needed is not in the //SYSEXEC concatenation.

Each command entered with a prefix of ! must stand alone. These commands cannot request more input from the operator, and they cannot see commands entered after they were entered through this facility.

Note: Commands cannot issue a TSO GETLINE or TGET request. A command procedure could issue an OPSWTO command with the REPLY operand to prompt the operator through a standard z/OS WTOR, but such command procedures are best run through the OPSECF facility of the ECF.

For more information about the CA OPS/MVS Server Facility, see the *Administration Guide*.

Commands Processed by the ECF

When the ECF sees a command that begins with a question mark, it routes the command to the OPSECF address space associated with the z/OS console from which the command was entered.

Note: A question mark is the default ECF character string. It can be changed to any other 1- to 3-character string (other than the current OSF command character string) with the OPSPRM OPS/REXX function or the OPSPARM command. For additional information, see the ECFCHAR parameter in the *Parameter Reference*.

OPSECF address spaces are created when a ?LOGON command is entered from a z/OS console, so ?LOGON must be the first question mark command entered.

TSO commands, TSO CLISTs, and OPS/REXX procedures run in OPSECF address spaces by being invoked after a ?LOGON can obtain additional input through the TSO GETLINE service routine (and its OPS/REXX equivalent), and any output they generate is routed back to the requesting console by the ECF.

Restrictions on TSO Command Runs

Neither commands run in an OPSOSF address space nor those run in an OPSECF address space can use TGET (they will hang) or TPUT (their output will be discarded). Full screen applications cannot be run under either mechanism; only line mode commands will work.

ECF Operation

The ECF is part of the CA OPS/MVS base product, so it requires no separate installation. The ECF is controlled by a number of product parameters, which are set with the OPSPRM OPS/REXX function or the OPSPARM TSO command. For more information about ECF installation and control, see the Administration *Guide*.

As with any part of z/OS, there are operational aspects to the ECF. These are described in the *Administration Guide*. For information about the ADDRESS OPSCTL command, see the *Command and Function Reference*.

Log On the ECF

You log onto the ECF by issuing a ?LOGON command to create an OPSECF address space to conduct a session with you by entering the following command at a z/OS (or JES3) console:

```
?LOGON userid/password
```

The *userid* specified must be a valid TSO user ID on the machine on which the command was entered, and *password* must be a valid password for that user ID. The ability of the ECF session to access data sets will be the same as if the logon had been to TSO.

The ?LOGON command shown above takes the CA OPS/MVS default for creating the OPSECF address space:

- If CA OPS/MVS itself started with SUB=JES2 (or JES3), then the OPSECF address space will be started with SUB=JES*x*. This is usually desirable, but if JES is down, or even if its health is suspect, then you should start your session with SUB=MSTR as described on the following page.
- If CA OPS/MVS itself started with SUB=MSTR, then the OPSECF address space attempts to start SUB=MSTR unless you code SUB(JES*x*) on your ?LOGON command.

  Note: CA OPS/MVS only allows SUB(MSTR) ECF sessions from a locally attached MCS console with Master authority; therefore you must code SUB(JESx) on your ?LOGON command if CA OPS/MVS is running SUB=MSTR and you are using any console other than the z/OS MCS console with master authority.
- If you log on using ?LOGON with SUB(MSTR), the ECFSECURITY parameter is ignored. This in effect sets the parameter to NOSECURITY and occurs whether the SUB(MSTR) is set explicitly or through starting CA OPS/MVS with SUB=MSTR.

ECFSECURITY is bypassed if the ECF is run in SUB=MSTR. In this case, the only security will be that defined for the ECF address space. Any ECF security defined in the initialization parameters during installation is ignored.

The most reliable means of running an ECF session is SUB=MSTR, even if CA OPS/MVS itself is being run SUB=JES3. You can use the following form of the ?LOGON command only from a locally attached MCS console with Master Authority:

```
?LOGON userid/password sub(mstr)
```

For this form of the command to work, the OPSECF JCL PROC must be in SYS1.PROCLIB and, if you are using DFSMS, all data sets referenced by that PROC must be cataloged in the z/OS master catalog.

You must use the ECF in this way to take advantage of its system rescue capability. When an OPSECF session is created SUB=MSTR, it has the security profile defined for the OPSECF address space. That is, it does not assume the security profile of the TSO ID with which you logged on, as it would if you had logged on SUB=JES*x*. This is so you can be sure to have a high enough data set access authority level to perform any system rescue that might be needed.

Conduct an Interactive ECF Session

Once you have successfully logged onto an ECF session from a console, you have a TSO session in READY mode with which you can interact. Output from your session will be routed back to your console automatically by CA OPS/MVS, but every line of input you enter for this session must be explicitly routed to it by prefixing it with a question mark (?). Do not use the OSF command character string. It is reserved for routing single-line requests to OSF TSO servers. This can be confusing at first because the OSF is still available, even while you are using the ECF. Worse, many commands you might enter through the ECF will also work if entered through the OSF, adding to the potential confusion.

This is an example session with the ECF to fix a broken VTAM procedure:

```
?LOGON syssup02/secret sub(mstr)
OPS1000I START OPSECF.SYSSUP02,SSID=OPSS,UID=SYSSUP02 START
<logon-related messages from your security product, if any>
OPS2106H SYSSUP02 LOGON FOR OPSS(1) AT 00:18:39 ON JANUARY 12, 2004
IKJ56644I NO VALID TSO user ID, DEFAULT USER ATTRIBUTES USED
READY
?edit 'sys1.proclib(vtam)' data
EDIT
?verify on
EDIT
?f '//vtamlst'
6000 //VTAMLST  DD   DSN=SYS1.VTAMLST,DISP+SHR
EDIT
?c 'disp+shr' 'disp=shr'
6000 //VTAMLST  DD   DSN=SYS1.VTAMLST,DISP=SHR
EDIT
?end save
READY
?LOGOFF
```

This use of the ECF is referred to as system rescue since it allows you to fix z/OS when it is in such a bad state that the typical tools you use (for example, TSO/ISPF) are not available. Naturally, the above session is somewhat simplified. Since it is happening at a console, unsolicited messages can interleave with the messages generated by the ECF session.

The IKJ56644I message above is typical and does not indicate any sort of error in the ECF session.

Log Off the ECF

You can logoff from an ECF session by entering the ?LOGOFF command from the console where the ECF session was started (from which the ?LOGON command was issued).

```
?LOGOFF
```

Recovery From Failures

ECF sessions can be prematurely terminated without a ?LOGOFF command having been issued under the following circumstances:

- The OPSECF address space is cancelled (ABEND S222).
- The OPSECF address space times out (ABEND S522).
- The OPSECF address space reaches CPU time limit (ABEND S322).
- A program running in the OPSECF address space incurs a system failure.

The ECF runs the TSO TMP in batch mode. The TMP will terminate when a command processor incurs a system ABEND. This is a limitation of the IBM TSO TMP, not of the ECF.

The ECF automatically detects such TSO address space failures and issues an automatic logoff from the console that started the TSO session. A ?LOGON command can be issued from the console to re-logon to the ECF.

Restrictions on TSO Commands Processed by the ECF and OSF

The following restrictions apply to TSO commands or CLISTs executed by the ECF, OSF TSO, OSF TSL, and OSF TSP:

- TGET/TPUT requests are ignored, since the TSO TMP is executed in batch mode.
- Only PUTLINE requests that typically would be printed to the SYSTSPRT file are returned by the ECF and OSF.
- Any GETLINE requests that typically would result in additional input being read from the SYSTSIN file will cause termination of the TSO command or CLIST being executed.
- In the ECF environment, the user PROFILE PREFIX value will not be set to the ID with which the operator logged on. Users of the ECF (OPSECF) facility and the OSF (OPSOSF) facility should run a PROFILE PREFIX(prefix) command before accessing anything but fully qualified data set names.
- LOGOFF commands should not be coded in CLISTs executed through the OSF (OPSOSF) facility. LOGOFF will terminate the OPSOSF address space, which the main CA OPS/MVS address space detects, and then immediately restarts it. If this happens often enough, the CA OPS/MVS address space stops restarting the servers.
- If CLISTs or REXX EXECs will be run through the OSF (OPSOSF) facility that issues long waits, tell CA OPS/MVS to start extra OPSOSF address spaces to make sure there are always some available to process operator and internal CA OPS/MVS requests or preferably schedule these CLISTs/REXX EXECs to run in OSF TSL servers that are intended for this purpose.

Security Considerations

Consider the following items regarding security for the ECF:

The OPSRMT and OPSCMD can potentially issue commands whose authority level checking is bypassed. When OPSRMT or OPSCMD issues a cross-system command, the OSF of the remote system using the authority levels assigned to the server address space eventually executes the command.

This same problem can arise when OPSRMT or OPSCMD is invoked to execute a TSO command on the local system. The command is ultimately executed by the OSF of the local system, and hence will be checked by your security system against the authority level of the server address space.

For this reason, you must secure the usage of the OPSRMT and OPSCMD TSO commands. This can be done in several ways:

- Read-protect the library in which the OPSRMT and OPSCMD load modules are placed.
- Use the OPUSEX exit routine to check the authority of the user when the OPSRMT or OPSCMD command is invoked. Both routines call this exit. The sample exit, as distributed, checks to make sure the caller has the TSO OPER privilege.
- Use CA ACF2, CA TopSecret, or RACF to restrict the use of OPSRMT and OPSCMD by user ID.
- Set the authority level of the server address spaces to the lowest common denominator of all users. This ensures that no user can gain more authority than he or she already has.

Note: CA OPS/MVS assigns all OSF server address spaces SUBMIT and OPER privileges by default.

Other Considerations

The ECFSECURITY parameter is bypassed if the ECF is run SUB=MSTR. This means that the only available security is that established for the ECF address space.

Note: Any ECF security defined in the initialization parameters is ignored.

Chapter 16: Multi-System Facility

This section contains the following topics:

Understanding the MSF

The Multi-System Facility(MSF) allows multiple copies of the product running on different z/OS images to communicate with each other through a variety of communication protocols. These z/OS images may be in a single data center, or they may be spread out around the world. Many of the CA OPS/MVS command processors, OPS/REXX host command environments, and OPS/REXX functions can perform their operations on any connected system. For example, when the MSF is installed and configured, you can perform any of these actions:

- Issue z/OS or subsystem commands to another system and retrieve the responses.
- Issue WTOs to consoles on other systems.
- Execute an OPS/REXX program in an OSF server address space that is on another system and retrieve the output.
- Issue SQL commands to retrieve or update rows in a relational table that resides on another system.
- Manage System State Manager resources on another system.
- Retrieve or update global variables that reside on another system.
- Browse the OPSLOG of another system.
- Control these CA OPS/MVS components on another system:
  - COF
  - MSF
  - OSF
- Issue commands and messages to CA Automation Point

The MSF supports the following communication protocols:

- LU 6.2
- XCF (through CAICCI)
- XES (through CAICCI)
- TCP/IP (through CAICCI)

MSF Support of JES3 (JES3 Only)

JES3 supports the routing of JES3 commands that are entered from a local processor. JES3 routes these commands to the global by using the JES3 PLEXSYN (syplex scope) CPF prefix. CA OPS/MVS submitted JES3 commands from a local processor will automatically make use of this facility.

However, there are cases where CA OPS/MVS users need to route a JES3 command from a local in one JESPLEX to JES3 a global in a different JESPLEX. Or, perhaps they do not have the JES3 PLEXSYN prefixes in place to do the command routing. For these cases, CA OPS/MVS can be configured to use MSF along with the JES3SYSID parameter. CA OPS/MVS will route JES3 commands entered on the local via MSF to the global MSF name that is defined in the JES3SYSID.

MSF Terminology

The MSF is not all that complicated, but since it uses network facilities, some of the following terms may be unfamiliar to you:

System

A *system* is equivalent to a VTAM domain. It usually consists of one CPU running one copy of VTAM (the SSCP) that controls all VTAM (network) resources connected to that CPU. A copy of CA OPS/MVS must be running on each system. A unique *system identifier* (*sysid*) identifies each system using the MSF.

MSF session

A logical connection through a VTAM session between two systems running CA OPS/MVS. Two copies of CA OPS/MVS can communicate if an MSF session exists between them.

Accessible

A system is said to be *accessible* to another system if an MSF session has been established between the two MSFs in each system.

Cross-system command

A command that is issued on one system and passed on to another system for processing through the MSF.

Local system

The system to which your terminal or console is connected if you are using MSF facilities interactively. If a program is using the MSF, then the local system is the one on which the program is running.

Remote system

Any system other than the local system.

Note: An MSF system can only communicate with other MSF systems with which it has established a session. For example, if systems A and B have established a session, and systems B and C have established a session, systems A and C will not be able to communicate with each other (unless a session has also been established between systems A and C).

MSF Installation

The installation of the MSF is covered in the *Installation Guide*. The discussion in that guide covers not only putting in MSF code, but also coding the parameters necessary to define the other copies of CA OPS/MVS with which the MSF will be communicating, as well as the system on which the MSF runs (the local system).

The MSF and CAICCI

The MSF supports the cross-system communication services provided by CAICCI. CAICCI, or the CAI Common Communications Interface, is one of the CCS for z/OS. It is a communications facility that allows CA solutions to communicate with one another. For information on the CCS for z/OS component that is required to run CAICCI, see the appendix “CCS for z/OS Component Requirements” in the *Installation Guide*.

For information about setting up MSF connections using CAICCI, see the *Administration Guide*. Review your CCS for z/OS documentation for information about installing CAICCI, and for further details about its features and functions.

MSF Operation

The MSF is a CA OPS/MVS component that cannot be started automatically when CA OPS/MVS is started and left to run on its own. This is because the MSF communicates with other systems that may or may not be up when your particular MSF initializes, and then may or may not stay up.

The following discussion assumes that the MSF is being operated manually. You are encouraged, however, to make the maximum use of the CA OPS/MVS AOF feature to automate any MSF operational tasks that you perform manually on a regular basis.

Activate the MSF VTAM APPLID

If you are using the LU 6.2 protocol, the MSF VTAM APPLID should be active at the time the MSF attempts to establish sessions with any other (all) CA OPS/MVS systems in your network. APPLID activation will usually be done at VTAM initialization time. If you defer activation until after VTAM initialization is complete, you can use the following VTAM operator command when you want to activate the MSF APPLID:

```
VARY NET,ACT,ID=msfname
```

The *msfname* value is the name of the member in SYS1.VTAMLST where the APPL statement of the MSF is stored.

Note: The member often has the same name as that of the APPL definition it contains, but VTAM does not require this.

The MSF APPLID must be active on *each* system on which CA OPS/MVS is running.

To activate the MSF VTAM APPLID

Assuming that you have three systems, System A, System B, and System C, and the SYS1.VTAMLST on each system contains a member defining the MSF APPLID for that system (assume the names are OPSMVSA, OPSMVSB and OPSMVSC respectively), you can activate the MSF application names with the following operator commands:

```
VARY NET,ACT,ID=OPSMVSA    issued on system A
VARY NET,ACT,ID=OPSMVSB    issued on system B
VARY NET,ACT,ID=OPSMVSC    issued on system C
```

You can use the AOF to issue these commands at the end of VTAM initialization, or use OPSVIEW (logged on to each system in turn) to issue them.

Start the MSF

The MSF must be active on all your z/OS systems before you can use it to issue cross-system commands. Starting the MSF requires that you perform the following steps (usually automatically rather than manually) on each of your systems.

To start the MSF

1. Start the CA OPS/MVS main product address space. This is usually done through the appropriate COMMND*nn* member of the Logical Parmlib Concatenation.
2. Tell CA OPS/MVS the name to use for the local system (the one on which it is running).

   The MSF picks up this parameter and uses it during MSF initialization. Setting this name, the SYSID of the local system, is usually done with an OPSPRM OPS/REXX function executed from the OPSSPA*nn* member of the Logical Parmlib Concatenation during CA OPS/MVS initialization.
3. Define the systems to the MSF (using ADDRESS OPSCTL MSF).
4. Start the local MSF (using ADDRESS OPSCTL MSF).
5. Activate the MSF sessions to remote systems (using ADDRESS OPSCTL MSF).

Each initialization step is explained in detail in the sections that follow.

Starting the System Task

Instructions for starting the CA OPS/MVS system task appear in the *Administration Guide*.

Set the Local System Identifier

You can set the local system identifier, the SYSID parameter, by using the OPSPRM OPS/REXX function from an OPS/REXX program.

Use this form of the OPSPRM function:

```
var = OPSPRM("SET","SYSID","value"[,,"system"])
```

For detailed descriptions of the arguments for the OPSPRM function, see the *Parameter Reference*.

Define Systems to the MSF

You must define the system on which the MSF is running (the local system) and all other systems with which it can communicate (the remote systems).

To define systems to the MSF

- Use OPSVIEW option 4.2

or

- You can issue the MSF DEFINE command through one of these methods:
  - From the OPS/REXX program specified in the AOFINIT REXX parameter
  - In the OPSTART2 OPS/REXX program
  - From an OPS/REXX program or an AOF rule
  - From a console using the OPSMSF command rule in the sample libraries

  The syntax for the MSF DEFINE command is:

  ```
  ADDRESS OPSCTL "MSF DEFINE keywords"
  ```

For detailed descriptions of the keywords for the MSF DEFINE command, see the *Command and Function Reference*.

Start Cross-system Sessions

Before MSF sessions can be established with a remote CA OPS/MVS system, you must first start MSF processing on the local CA OPS/MVS system. When using the LU 6.2 protocol, the MSF responds by opening its VTAM ACB.

To start cross-system sessions

Issue the MSF START command through one of these methods:

- In the AOF
- From an OPS/REXX program
- From a console using the OPSMSF command rule in the sample libraries

Note: You can issue the MSF START command only *after* the local system has been defined using the MSF DEFINE command described in the previous section.

The syntax for the MSF START command is as follows:

```
ADDRESS OPSCTL "MSF START keywords"
```

For detailed descriptions of the keywords for the MSF START command, see the *Command and Function Reference*.

Activate MSF Sessions to Remote Systems

Once MSF processing has been started on the local system (the one on which the CA OPS/MVS system you are dealing with is running), you can tell the MSF to initiate sessions with all the other CA OPS/MVS systems you have defined to it (with previous MSF DEFINE statements).

To activate MSF sessions to remote systems

1. Issue one or more MSF ACTIVATE commands through one of these methods:
   - In the AOF
   - From an OPS/REXX program
   - From a console using the OPSMSF command rule in the sample libraries

   When using the LU 6.2 protocol, the MSF responds by issuing VTAM OPNDST to establish the session requests.

The syntax for the MSF ACTIVATE command is as follows:

```
ADDRESS OPSCTL "MSF ACTIVATE keywords"
```

For detailed descriptions of the keywords for the MSF ACTIVATE command, see the *Command and Function Reference*.

Define and Activate MSF Sessions

Assume you need to define and activate your complex of three domains with VTAM application names already set up (OPSMVSA, OPSMVSB, and OPSMVSC) in each domain with MSF system IDs SYSA, SYSB, and SYSC.

To define and activate the MSF sessions

- Issue the following set of commands on all systems once CA OPS/MVS has completed its initialization:

  ```
  ADDRESS OPSCTL "MSF DEFINE MSFID(SYSA) APPLID(OPSMVSA)"
  ADDRESS OPSCTL "MSF DEFINE MSFID(SYSB) APPLID(OPSMVSB)"
  ADDRESS OPSCTL "MSF DEFINE MSFID(SYSC) APPLID(OPSMVSC)"
  ADDRESS OPSCTL "MSF START NORETRY"
  ADDRESS OPSCTL "MSF ACTIVATE MSFID(ALL)"
  ```

  Usually, you issue these commands in the OPSTART2 OPS/REXX program. CA OPS/MVS runs the OPSTART2 OPS/REXX program automatically immediately after it finishes initialization. For more information about this program, see the *Administration Guide*.

- However, you can also issue the commands through one of these methods:
  - In the AOF
  - From an OPS/REXX program
  - From a console using the OPSMSF command rule in the sample libraries

  Exactly the *same* parameters can be used on all three systems. This is because the MSF learns which one of the three systems defined is the local system from the specification through OPSPRM and does different ACTIVATE processing for the local system (the MSF OPENs the VTAM ACB) than for the remote systems (the MSF executes OPNDSTs to start sessions with them). The parameters of the MSF were purposely designed this way to reduce the number of parameters that must be set.

Auto-connecting MSF Sessions

As soon as the MSF START command has been issued, other systems can establish MSF sessions with the local system. This is true even when no other remote systems have been defined to the local system. An undefined remote system that tries to establish a session with the local system will automatically be defined in the local system as if an MSF DEFINE command of the following form was actually issued on the local system:

```
ADDRESS OPSCTL "MSF DEFINE MSFID(sysname) APPLID(vtamname) NORETRY"
```

The *sysname* operand is passed by the remote system to the local system when the MSF session is established. When using the LU 6.2 protocol, the *vtamname* operand is the application name by which the VTAM domain of the local system identifies the VTAM application of the remote system.

Display Systems and MSF Sessions

You can display the status of the local system and remote sessions by issuing the MSF LIST command.

Note: For detailed descriptions of the keywords for the MSF LIST command, see the *Command and Function Reference*.

To display systems and MSF sessions

1. Issue the MSF LIST command through either of these methods:
   - From an OPS/REXX program
   - From a console using the OPSMSF command rule in the sample libraries
2. The syntax for the MSF LIST command is as follows:

   ```
   ADDRESS OPSCTL "MSF LIST keywords"
   ```

The following is a sample display of the information returned by the MSF LIST command:

```
Local     Status   Applid    VTAM   Retry  Max  Current    VTAM
System   VTAM      VTAM   Password Secs  Retry  Retry  Open Error
-------- -------- -------- -------- ----- ----- ------- -----------
SYSA     ACTIVE   OPSMVSA           30          0

---------------------------- Remote System---------------------------------
     System                     Delay   Retry  Max  Current    VTAM
Sel  Name     Status   APPLID   Value   Secs  Retry  Retry  Rtncd Fdbk2 Type
---  -------- -------- -------- -------- ----- ----- ------- ----------- -----
     SYSB     ACTIVE   OPSMVSB     1     30    5     0       X'00' X'00' APPC
     SYSC     INACTIVE OPSMVSC     1     30    5     1       X'00' X'00' APPC
     SYSD     FAILED   OPSMVSD     1        NO RETRY         X'08' X'01' APPC
     SYSE     INACTIVE OPSMVSF     1     30    4     4       X'00' X'00' APPC
     SYSF     FAILED   OPSMVSF     30    30    4     2       X'08' X'01' APPC
```

In the above display:

- The local system is called SYSA.
- Five remote systems have been defined.
- Only one MSF session from the local system is active (the session to system SYSB).
- The MSF session from SYSA to SYSC has been deactivated using the MSF DEACTIVATE command.
- The MSF session from SYSA to SYSD has not been established because it was activated with the NORETRY keyword and the remote system (SYSD) has not been able to establish the MSF session.
- Both sessions to SYSE and SYSF failed. The session to SYSE was activated in RETRY mode, but has reached the maximum number of RETRYs and must be reestablished by the MSF on the SYSE system.

- The session to SYSF is using RETRY mode, with retries every 30 seconds and a maximum of 4 retries. The MSF is currently retrying for the second time to reestablish the session.

Like any part of the z/OS environment, the MSF can operate with the CA OPS/MVS AOF. Thus, if the NORETRY and RETRY options supported by the MSF cannot establish MSF sessions and keep them up in your environment, you can program more sophisticated procedures using the same AOF facilities with which you control all the other systems in your environment.

Issue Commands to Remote Systems

You can issue commands to other systems using either of the following commands:

- OPSRMT command

  Use the OPSRMT command to issue any TSO command to another system (including the OPSCMD command).

- OPSCMD command

  Use the OPSCMD command to issue any operator command (z/OS or JES) to another system.

Issue Cross-system TSO Commands

You can issue TSO commands to other systems by issuing the OPSRMT command (alias OR). This command enables you to issue TSO commands to other systems from either a TSO terminal or a console logged into the CA OPS/MVS ECF component. For the syntax of the OPSRMT command, see the *Command and Function Reference*.

Issuing the OPSRMT command without specifying command text causes the OPSRMT command processor to enter subcommand mode. The CA OPS/MVS product routes all TSO commands you enter while in subcommand mode to the subsystem you specified on the OPSRMT command.

When the TSO command immediately follows the system ID on an OPSRMT command, CA OPS/MVS considers all text beyond the system ID to be part of the command text. For example, in the following example, the TSO command text is LISTCAT LEVEL(SYS1):

```
OR SYSTEM3 LISTCAT LEVEL(SYS1)
```

Note: When the MSF is not installed or is not active on the local system, the OPSRMT command returns an error message when you specify any system ID other than that of the local system (or *).

The OPSRMT command with the system ID of the local system (or *) is always valid as long as the CA OPS/MVS address space is active. The MSF does not need to be installed or active when specifying the local system ID on the OPSRMT command.

Issue JES3 Commands

When you issue a JES3 command from a local processor using OPSCMD or ADDRESS OPER, CA OPS/MVS by default uses the JES3 PLEXSYN (sysplex scope) CPF prefix to route the command to the global system in the JESPLEX and return the output.

When the JES3SYSID parameter is set, MSF connections are use to route the JES3 command from the JES3 local system to the JES3 global system. The global here refers to the MSF SYSID the JES3 global system as defined by the JES3SYSID parameter.

The JES3SYSID parameter should only be defined if either of the following are true:

- There is no sysplex scope JES3 PLEXSYN prefix defined.
- The JESPLEXes are in different sysplexes and there is a need to route JES3 commands from a local to a global that is in a different sysplex.

OPSSEND Function and ADDRESS WTO—Pass Messages to Remote Systems

WTO messages issued on one system can be passed to another system through the OPSSEND function or the ADDRESS WTO instruction.

OPSSEND Function

The OPSSEND function can only be called from in an AOF rule; it sends the message intercepted by the rule to a remote system.

The OPSSEND function has the following format:

If the following AOF rule is enabled on system SYSA, all messages whose message ID starts with IEF will be sent to system SYSB:

```
)MSG IEF*
)PROC
CALL OPSSEND 'SYSB' 'B'
)END
```

The first operand passed to the OPSSEND function call is the target system ID.

ADDRESS WTO Instruction

You can also use the ADDRESS WTO instruction to issue messages from any OPS/REXX program to a remote system.

The ADDRESS WTO instruction has the following format:

```
ADDRESS WTO "TEXT('messagetext') keywords"
```

For more information about the ADDRESS WTO instruction, see the *Command and Function Reference*.

Important! Although you can use both the OPSSEND function and the ADDRESS WTO instruction to pass messages to remote systems, we recommend that you use the ADDRESS WTO instruction.

Shut Down MSF Sessions and Systems

To shut down the MSF, do the following:

1. Deactivate all active MSF sessions.
2. Stop the local MSF.

These tasks are described in this section.

Deactivate MSF Sessions

You can deactivate an MSF session between the local system and any remote system.

To deactivate your MSF session

1. Issue the MSF DEACTIVATE command through either of these methods:
   - From an OPS/REXX program
   - From a console using the OPSMSF command rule in the sample libraries
2. Use the following syntax for the MSF DEACTIVATE command:

   ```
   ADDRESS OPSCTL "MSF DEACTIVATE keywords"
   ```

   For detailed descriptions of the keywords for the MSF DEACTIVATE command, see the *Command and Function Reference*.

After an MSF session is deactivated, you cannot issue cross-system commands between the two systems involved in the MSF session, and unsolicited messages are not passed between the two systems. If any commands were in progress, they are aborted with a message showing that the MSF session was deactivated.

Stop the MSF

You can stop the local MSF.

To stop the MSF

1. Issue the MSF STOP command through either of these methods:
   - From an OPS/REXX program
   - From a console using the OPSMSF command rule in the sample libraries
2. Use the following syntax for the MSF STOP command:

   ```
   ADDRESS OPSCTL "MSF STOP keywords"
   ```

   For detailed descriptions of the keywords for the MSF STOP command, see the *Command and Function Reference*.

After the local MSF is stopped, you cannot issue any cross-system commands or messages from or to the local system until an MSF START command has restarted the local MSF. If any commands were in progress, they are aborted with a message showing that the MSF session was stopped.

Remove System Definitions

You can delete the definition of a system (previously added with an MSF DEFINE command).

To remove system definitions

1. Issue the MSF DELETE command through either of these methods:
   - From an OPS/REXX program
   - From a console using the OPSMSF command rule in the sample libraries
2. Use the following syntax for the MSF DELETE command:

   ```
   ADDRESS OPSCTL "MSF DELETE keywords"
   ```

   For detailed descriptions of the keywords for the MSF DELETE command, see the *Command and Function Reference*.

A system that has an active MSF session cannot be deleted until you deactivate the MSF session through the MSF DEACTIVATE command. You cannot delete the definition of the local system until an MSF STOP command has been issued.

Recovery from Failures

The MSF distinguishes between these types of failures:

- MSF system failures
- MSF session failures
- Cross-system command failures

The failures are described in the sections that follow.

MSF System Failures

When the MSF on a particular system is unable to continue, all sessions to and from that system will fail. Any of the following conditions may cause an MSF system failure:

- An operating system crash
- A VTAM shutdown
- Deactivation of the CA OPS/MVS VTAM application definition
- Failure of the CA OPS/MVS address space

Any remaining MSF systems can try to reestablish their sessions with the failing system, depending on how the failing system was defined. If the failing system was defined or activated with the RETRY option, the other system will try to reestablish the session. If it was defined or activated with the NORETRY option, the other system waits for the failing system to reestablish the session.

Note: This occurs after the MSF has been restarted and the session is restarted with the RETRY option on the failing system.

MSF Session Failures

When a session between two MSF systems fails, all communication between the two MSF systems halts.

Any of the following conditions can cause an MSF session failure:

- Two things can cause a session failure on the local system:
  - MSF system failure on a remote system
  - Deactivation of the VTAM application definition of a remote system
- Sessions will fail on both systems involved in the session if the communications link between two systems is disrupted (a lost phone connection, a lost satellite link, and so on).

After an MSF session failure:

- You cannot issue cross-system commands between the two systems involved in the MSF session.
- Unsolicited messages are no longer passed between the two systems.
- Any commands that were in progress will be aborted with a message showing that the MSF session was deactivated.
- Systems that have defined the session with the RETRY option automatically try to reestablish the session (assuming that both systems involved in the session are still active).

If possible, the MSF writes messages to the system log to identify the commands, messages, or both, that were being processed when a session failed.

Cross-system Command Failures

A cross-system command can fail for a variety of reasons:

- An MSF system failure
- An MSF session failure
- An excessive number of messages are returned in response to the command
- The command takes an excessive amount of time to complete
- The z/OS (sub)system that will process the command fails

Except for the first two causes, the MSF automatically recovers from cross-system command failures.

In most cases, the command issuer receives a message to show the specific cause of the command failure; for example, session deactivated, VTAM shutdown, excessive output. If the originating OPSRMT command was issued from a TSO CLIST, the return code from OPSRMT also indicates the type of failure.

If possible, the MSF also writes a message to the system log to identify a failing cross-system command.

Security Considerations

Consider the items described in the following sections when establishing security for the MSF.

VTAM APPLIDs

Any VTAM application can establish an MSF session with CA OPS/MVS. Assuming that the correct VTAM protocols are used, nothing prevents another VTAM application from introducing commands (TSO or operator commands) into the MSF by emulating a remote system.

For this reason, it is *critical* to protect the use of VTAM application names using the PASSWORD operand on the VTAM APPL statement defining *each* application name.

For this same reason, you must read-protect your VTAM definition library so that unauthorized users cannot read these passwords.

OPSRMT and OPSCMD TSO Commands

OPSRMT and OPSCMD can issue commands whose authority level checking is bypassed. When OPSRMT or OPSCMD issues a cross-system command, the command eventually executes on the remote system using the authority levels assigned to the CA OPS/MVS address space.

This same problem can arise when OPSRMT or OPSCMD is invoked with the name of the local system. The command executes in the CA OPS/MVS address space of the local system and will be checked by your security system against the authority level of the CA OPS/MVS address space.

In general, OPSRMT executes the TSO command on the target system using the same security that the issuing user would have, had he or she logged onto TSO on the target system manually and issued the TSO command from a session. Specifically, before each command executes on the target system, CA OPS/MVS sets up the RACF, CA Top Secret, or CA ACF2 environment to ensure that the security products use the security clearance of the issuing user.

The security system on the target system must know the user ID of the user issuing a remote command *and* the user must have the same user ID on the target system. If this is not the case, a user can bypass security checks (and password checking of user IDs).

For example, suppose that user ID ABC is known on two MSF systems (called SYS1 and SYS2) but the user ID belongs to user X on SYS1 and user Y on SYS2. User X can log on to system SYS1 and issue an OPSRMT command to SYS2. The command will execute on the SYS2 system using the security clearance of user Y, without user X ever knowing the password of user Y.

You can install additional security beyond that available for your security package using any of these methods:

- Read-protecting the library in which the OPSRMT and OPSCMD load modules are placed.
- Using the OPUSEX exit routine to check the authority of the user when the OPSRMT or OPSCMD command is invoked. Both routines call this exit. The sample exit, as distributed, checks to make sure the caller has the TSO OPER privilege.
- Writing security rules.
- Using CA ACF2, CA Top Secret, or RACF to restrict the use of OPSRMT and OPSCMD. Note that the OPSECURE OPS/REXX function can be used in security rules to check the authority of the user to issue these commands.

Security for Other Cross-system Operations

There are two categories of cross-system operations:

- Display- or query-type operations that do not affect the state of the system or CA OPS/MVS. An example of this type of operation is the use of the cross-system OPSLOG Browse facility.
- Update-type operations that alter the state of the system or CA OPS/MVS. An example of this type of operation is a cross-system ADDRESS SQL UPDATE.

When you define the remote systems on each system, you specify whether each remote system is considered to be a SECURE system. SECURE remote systems can send *both* query- and update-type operations to this system. Systems that are defined as NOSECURE systems can only send display- or query-type operations to this system. For example, if you connect a non-secure test system to your production systems through the MSF, you should specify the NOSECURE keyword on the ADDRESS OPSCTL MSF DEFINE statement when defining the test system as a remote MSF system to each of the production systems.

Chapter 17: Expert System Interface

This section contains the following topics:

Overview

You can use the Expert System Interface(ESI), or OPSLINK programming interface, to gain access to some CA OPS/MVS facilities from an application written in a high-level language or in assembler language. Specifically, you can use OPSLINK to:

- Execute operator commands.
- Execute TSO commands (if running under the TSO TMP interactively or in batch).
- Access CA OPS/MVS global variables, update them, or both.

This programming interface is implemented as a subroutine load module also named OPSLINK.

Note: The INITESI parameter must be set to YES, if you are licensed for and are using the ESI. When the INITESI parameter is set to NO, most ESI requests will fail and the following message will appear in the OPSLOG/SYSLOG:

```
OPS9526H ESI FEATURE HAS NOT BEEN LICENSED
```

Calling Language Dependencies

You can call OPSLINK from any language that supports the calling sequence described in the following sections. The rest of this chapter describes in detail how to call OPSLINK from PL/1, COBOL, and assembler language routines.

Call OPSLINK from PL/1 Programs

The arguments passed to OPSLINK are either text strings or integer values. All text strings passed should be defined in the calling programs as CHAR(*length*) VARYING. All integers passed should be defined as FIXED BIN(31).

Important! Failure to observe these definition requirements may result in obscure storage overlay problems, S0Cx type abends, or both.

Define an Output Array in PL/1 Programs

The optional sixth argument of all OPSLINK calls is an output array that should be defined as:

```
DCL ARRAY(1000) CHAR(256) VARYING;
```

The size 1000 is arbitrary, but should be large enough to accommodate the maximum number of output lines expected from each call. The argument passed to OPSLINK before the sixth argument sets the size of the array (1000 in the above example).

Define the OPSLINK Routine in PL/1 Programs

The OPSLINK routine should be defined as follows:

```
DCL OPSLINK ENTRY OPTIONS(ASM,INTER,RETCODE);
```

Before the first call is made to OPSLINK, you must load the subroutine into memory by issuing the FETCH PL/1 instruction, as follows:

```
FETCH OPSLINK;
```

After the last call is made to OPSLINK, unload the subroutine from memory by issuing the RELEASE PL/1 instruction, as follows:

```
RELEASE OPSLINK;
```

OPSLINK returns a return code in PL/1 variable RETCODE. Return codes from OPSLINK and their meanings are explained at the end of this section.

Call OPSLINK from COBOL Programs

The arguments passed to OPSLINK are either text strings or integer values. All text strings passed should be defined in the calling programs as follows:

```
DATA DIVISION.
01 ARGSTR.
   03 STRING-LENGTH   PIC 9(4) COMP VALUE 0.
   03 STRING-TEXT     PIC X(256) VALUE SPACES.
```

The STRING-LENGTH should be set to the number of significant characters in STRING-TEXT. Zero indicates a null string.

All integers passed should be defined in the calling program as:

```
DATA DIVISION.
01 ARGINT             PIC 9(8) COMP.
```

Important! Failure to observe these definition requirements may result in obscure storage overlay problems, S0Cx type abends, or both.

Define an Output Array in COBOL Programs

The optional sixth argument of all OPSLINK calls is an output array that should be defined as:

```
DATA DIVISION.
   01 ARRAY          OCCURS 200 TIMES.
   03 LINE-LENGTH    PIC 9(4) COMP.
   03 LINE-TEXT      PIC X(256).
```

The size 200 is arbitrary, but should be large enough to accommodate the maximum number of output lines you expect each call to return. The fifth argument passed to OPSLINK sets the size of the array.

Note: The fifth argument had set it to 200 in the above example.

Define the OPSLINK Routine in COBOL Programs

The OPSLINK routine should be defined as follows:

```
DATA DIVISION.
 77  OPSLINK              PIC X(8) VALUE 'OPSLINK'.
 77  RET-CODE             PIC X(4) VALUE SPACES.
```

OPSLINK returns a return code in COBOL variable RETCODE. Return codes from OPSLINK and their meaning are explained at the end of this section.

Call OPSLINK from Assembler Programs

The arguments passed to OPSLINK are either text strings or integer values. All text strings passed should be defined in the calling program as follows:

```
ARGSTR DC H'length',C'text'
```

The *length* indicates the number of characters in *text*. Zero indicates a null string. All integer values are fullword signed binary and should be defined as follows:

```
ARGWORD DC F'value'
```

The optional 6th argument of all OPSLINK calls is an output array that should be defined as:

```
ARRAY  DS 1000CL258
```

The number 1000 is arbitrary but should be large enough to accommodate the maximum number of output lines expected to be returned by each call. The fifth argument passed to OPSLINK sets the size of the array. The size was set to 1000 in the above example. Each element is 258 bytes long, starting with a halfword-length field, followed by a 256-byte string area.

Assembler programs must turn on the VL bit when calling OPSLINK. The VL bit is the high-order bit in the last word of the parameter list. Also, the calling program must have an AMODE of 31 if a LOAD followed by either a BALR or a BASR is issued.

Before the first call is made to OPSLINK, you must load the subroutine into memory by issuing the LOAD macro instruction. After the last call to OPSLINK, delete the subroutine from memory using the DELETE macro instruction. On return from the LOAD macro, if register 15 is zero, register 0 contains the address of the load module in bits 1-31. Check the appropriate IBM manual for further details on the LOAD and DELETE macros.

The CA OPS/MVS API calling sequence from assembler is as follows:

```
         LOAD  EP=OPSLINK      Load the subroutine
         LTR   R15,R15
         JNZ   ERROR
         ST    R0,OPSLINKA     Save the subroutine address          .
          .
          .
         OC    PARMLAST(1),=X'80'  Turn on VL bit
         LA    R1,PARMLIST     Parameter list address
         L     R15,OPSLINKA    Load OPSLINK address
         BASR  R14,R15         Call OPSLINK
          .
          .
          .
         DELETE EP=OPSLINK     Delete OPSLINK
ERROR    DS    0H
          .
          .

OPSLINKA DS    A               Address of OPSLINK
*
PARMLIST DS    0A              Parameter list passed to OPSLINK
         DC    A(SUBFUNC)      ARG1
         DC    A(SYSID)        ARG2
         DC    A(SUBSYS)       ARG3
         DC    A(CMDSTR)       ARG4
         DC    A(LINECNT)      ARG5
         DC    A(ARRAY)        ARG6
         DC    A(IMSID)        ARG7
PARMLAST DC    A(MFORM)        ARG8
*
SUBFUNC  DC    H'6',C'OPSTSO'  Issue TSO command
SYSID    DC    H'0',C' '       No system ID
SUBSYS   DC    H'0',C' '       No OPS/MVS subsystem name
CMDSTR   DC    H'5',C'LISTC'   TSO command
IMSID    DC    H'0',C' '       No IMSID
MFORM    DC    H'1',C'M'       MFORM(M)
LINECNT  DC    F'100'          Allow for 100 lines
ARRAY    DS    H,CL256         Room for 1 Line
         DS    99CL258         Room for 99 more lines
```

OPSLINK returns a return code in register 15. Return codes from OPSLINK and their meaning are explained at the end of this section.

OPSLINK Function Calls

The OPSLINK function can be called for three different purposes. The first argument passed to OPSLINK specifies which of these three functions you want to use. Valid values for the first argument are:

- OPSTSO (to execute a TSO command)
- OPSCMD (to execute a system command)
- OPSGLOBL (to access, update, or both CA OPS/MVS global variables)

The following sections describe each subfunction call in detail.

Note: PL/1 calling syntax is used to describe the arguments. For COBOL and assembler calling routines, you must convert this syntax to that appropriate for the calling language (as described in the preceding sections).

Execute TSO Commands

You can execute TSO commands by calling OPSLINK with the following arguments:

```
CALL OPSLINK('OPSTSO',
             SYSID,
             SUBSYS,
             CMDSTR
             [,LINECOUNT
             [,ARRAY
             [,DDNAME ]]])
```

You can execute TSO commands only if the calling program is running under the TSO Terminal Monitor Program (TMP) interactively or in batch.

CALL OPSLINK statements that execute TSO commands can have these arguments:

SYSID

The eight-character ID of the system where the TSO command should execute. If you specify a value of all blanks or all binary zeroes, the local system is assumed (the one on which the calling program is running). The name used will be the uppercased version of the one passed.

Note: This parameter is currently ignored.

SUBSYS

The four-character subsystem name of the target CA OPS/MVS subsystem that should process this request. The name used will be the uppercased version of the one passed.

Note: This parameter is currently ignored.

CMDSTR

The TSO command to be executed. The command can include a leading percent sign (for implicit execution of CLISTs). The maximum length of the command string is 256 characters.

LINECOUNT

A fullword binary value that indicates the maximum number of output lines to be returned in the ARRAY argument (see below). The value should be non-negative. If zero is specified, the default is used. If you specify the LINECOUNT parameter, you should also specify the ARRAY parameter. The number of lines actually placed in the output array by OPSLINK is returned in this field.

You should *not* use a constant in the calling sequence; instead, use the name of a program variable for this argument. Also, before each subsequent OPSLINK call, you must reinitialize this variable to the maximum number of lines that the array can hold.

ARRAY

A character string array in which the output from the request is returned.

DDNAME

An eight-character string specifying the ddname to which output will be redirected before it is placed in the output array. This operand is optional and if omitted will result in dynamic allocation of a VIO data set to hold the TSO command output.

If you intend to issue a sequence of TSO commands, you can improve performance by pre-allocating a VIO (or disk) data set and passing its ddname to OPSLINK. The ddname actually used by OPSLINK is returned in this field.

You should *not* use a constant in the calling sequence; instead, use the name of a program variable for this argument.

Execute Operator Commands

You can execute JES, VM, and z/OS operator commands as well as any command that can be issued using the OPSCMD TSO command by calling OPSLINK with the following arguments:

```
CALL OPSLINK('OPSCMD',
             SYSID,
             SUBSYS,
             CMDSTR
             [,LINECOUNT
             [,ARRAY
             [,IMSID
             [,MFORM ]]]])
```

The SYSID and SUBSYS arguments have the same meanings as they do in CALL OPSLINK statements that execute TSO commands. The other arguments have these meanings:

CMDSTR

The system command to be executed. The command can be any command that can be issued using the OPSCMD command. The maximum length of the command string is 256 characters.

SYSID

The eight-character MSFID of the MSF system where the system command should execute. If you specify a value of all blanks, all binary zeroes or a single asterisk followed by seven blanks, the local system is assumed (the one on which the calling program is running). The name used will be the uppercased version of the one passed.

Note: The MSF feature must be installed and licensed if you specify anything other that the local system.

SUBSYS

The four-character subsystem name of the target CA OPS/MVS subsystem that should process this request. The name used will be the uppercased version of the one passed.

LINECOUNT

A fullword binary value that indicates the maximum number of output lines to be returned in the ARRAY argument. The value should be non-negative. If you specify zero, the default is used. If you specify the LINECOUNT parameter, you should also specify the ARRAY parameter.

The number of lines actually placed in the output array by OPSLINK is returned in this field. Instead of using a constant in the calling sequence, use the name of a program variable for this argument. Also, before each subsequent OPSLINK call, you must reinitialize this variable to the maximum number of lines that the array can hold.

ARRAY

A character string array in which the output from the request is returned.

IMSID

The IMSID of the IMS system where the command should be sent. The maximum length of the command string is four characters.

MFORM

A one-character operand that indicates the format of your output. If used, this operand must be either of the following:

- M, which specifies that the output be not stamped with the job name and number.
- J, which specifies that the output be stamped with this data.

Access and Update Global Variables

You can access or update global variables by calling OPSLINK with the following arguments:

```
CALL OPSLINK('OPSGLOBL',
             SYSID,
             SUBSYS,
             VARNAME
             [,LINECOUNT
             [,ARRAY
             [,OPTION
             [,VARVALUE
             [,OLDVALUE ]]]])
```

Note: This subfunction call is equivalent to the OPSVALUE() REXX function call to access/update global variables from in REXX programs and AOF rules. Results normally returned in the external data queue by the OPSVALUE() function are returned in the output ARRAY (if any) instead.

Arguments you can use in CALL OPSLINK statements that operate on global variables are:

SYSID

An eight-character system ID where the request should be executed. If you specify a value of all blanks or all binary zeroes, the local system is assumed (the one on which the calling program is running). The name used will be the uppercased version of the one passed.

Note: This parameter is currently ignored.

SUBSYS

The four-character subsystem name of the target CA OPS/MVS subsystem that should process this request. The name used will be the uppercased version of the one passed.

VARNAME

The name of the global variable. This name must start with a valid global variable prefix (GLOBAL or GLVTEMP). No substitution of qualifiers is made for the compound symbol. For additional information on global variables see the chapter titled *Global Variables Explained* in this book.

LINECOUNT

A fullword binary value that indicates the maximum number of output lines to be returned in the ARRAY argument (see below). The value should be non-negative. If zero is specified, the default is used. If you specify the LINECOUNT parameter, also specify the ARRAY parameter. The number of lines actually placed in the output array by OPSLINK is returned in this field.

Instead of using a constant in the calling sequence, use the name of a program variable for this argument. Also, before each subsequent OPSLINK call, you must reinitialize this variable to the maximum number of lines that the array can hold.

ARRAY

A character string array in which the output from the request is returned.

OPTION

A one-character string containing a code that represents the operation you want to perform on a global variable.

Note: The OPTION code is related to the equivalent OPS/REXX OPSVALUE function action code. For additional information, see the OPSVALUE documentation.

VARVALUE

A character string variable in which the current value of the specified global variable is returned (options O and V), or which contains the new value for a global variable (options A and U only). The string variable should be large enough to contain a string of 256 characters.

Instead of using a constant in the calling sequence, use the name of a program variable for this argument. Also, before each OPSLINK that retrieves a value (options O and V), you must reinitialize the length of the string to its maximum size.

OLDVALUE

A character string variable in which the current or old value of the specified global variable must be specified and is used for verification prior to update (option C). This argument must only be specified for option C.

Codes for the OPTION Argument

You can specify one of the following codes for the OPTION argument:

6

Delete single global variable

Removes the single global variable specified by VARNAME without removing any of its subnodes. After the operation VARVALUE contains 1 if the node was deleted or a 0 if the node was not found. ARRAY is not modified.

A

Add a value to a global variable

Adds a numeric value, specified by the VARVALUE argument to the existing global variable. If VARVALUE is not numeric the operation fails with return code 41. After the operation VARVALUE contains the sum of the global variable and the increment. ARRAY is not modified.

C

Compare and update

Updates a global variable after verifying its current value against the *oldvalue* provided. VARVALUE contains 1 if the update was successful or a 0 if the comparison failed. ARRAY is not modified.

D

Drop a global variable

Drops the global variable, see the REXX definition of DROP. The global variable still exists but its value is reset to its uninitialized value. This value is returned in VARVALUE. However, the L call (described below) still lists it. ARRAY is not modified.

E

Check existence of a global variable

Checks the global variable for existence. After the operation, VARVALUE contains one of the following global variable status value characters:

I-Initialized

U-Uninitialized

N-Does not exist

ARRAY is not modified.

I

Return information for a global variable

Returns information about all of the immediate subnodes of the global variable in the ARRAY. Two consecutive array entries are returned for each global variable node. The first entry contains the next segment of the global variable name and the second entry contains information about that global variable. See the OPSVALUE('I') OPS/REXX function for additional details. After the operation, VARVALUE contains the number of processed global variables.

J

Immediate subtree count

Returns a count of all the immediate subnodes of a global variable. ARRAY is not modified.

K

Subtree count

Returns a count of all the subnodes of a global variable. ARRAY is not modified.

L

List the name or names of global variables

List the names of all immediate subnodes for a partial global variable symbol. Returns information about all of the immediate subnodes of the global variable in the ARRAY. See the OPSVALUE('L') OPS/REXX function for additional details. After the operation, VARVALUE contains the number of global variables processed.

N

Return the value of a global variable

Returns a null string if the global variable does not exist. VARVALUE contains the value of a global variable.

ARRAY is not modified.

O

Obtain the value of a global variable

Returns the global variable value in VARVALUE. If global variable does not exist, it is not created, but an error is returned instead - Return code 40. ARRAY is not modified.

R

Remove a global variable

Deletes the global variable and all of its subnodes and the L call (described above) will no longer list it. VARVALUE contains the number of global variables deleted. ARRAY is not modified.

U

Update the value of a global variable

Updates the value of the global variable with the value specified in the VARVALUE argument (see below). ARRAY is not modified.

V

Return the value of a global variable

Returns the value of the global variable in VARVALUE. If the global variable does not exist, it is created, and its name is returned as the value. ARRAY is not modified.

Return Codes from OPSLINK

OPSLINK will return one of the following return codes. In most cases, an appropriate error message is also issued to explain the specific error.

0

Function executed successfully

4

Product or system service failed

12

TSO service routine error

16

TSO service routine ABEND

20

TSO service routine error

24

Output array overflow

80

TSO/E is not installed

88

Main product address space is not active

92

Parameter list error

100

Serious product control block error

104

Main product address space terminated

184

ABEND failure

Sample Programs that Use OPSLINK

To find sample programs that demonstrate the use of the ESI, see the appendix “Sample Programs” or preferably use your local scan utility (for example: the ISPF SRCHFOR utility) to scan the OPS/MVS sample data set for the text string OPSLINK.

Sample programs are provided written in C, COBOL and PL/1.

Chapter 18: CICS Operations Facility

This section contains the following topics:

COF Overview

The CICS Operations Facility(COF) enables CA OPS/MVS to control, operate, and administer one or more CICS regions.

The COF is an optional feature of CA OPS/MVS. Some CICS messages are written only to internal data sets called transient data queues. These messages are not broadcast through the normal z/OS subsystem message interface; therefore, AOF rule processing will *not* see them. Rather, the COF selectively traps the messages and forwards them to the AOF for rule processing.

The functionality provided by the COF is part of the integrated and unified solution for controlling z/OS subsystems such as CICS, JES, and IMS.

Install and Start the COF

To install the COF, you need only define a single CICS transaction and program using RDO. Once the COF is installed, you can start it automatically at CICS initialization through the PLT or a batch terminal.

In CA OPS/MVS, you can control the activation of the COF interface by setting the INITCOF and CICSAOF parameters to YES or NO. Once activated, you can then tailor the way in which the COF intercepts transient data queue messages through the ADDRESS OPSCTL COF commands or OPSVIEW option 4.12.

For complete instructions on how to install the COF, see the *Installation Guide*.

How You Can Use the COF

Here are some things you can do with the COF:

- If security and console definitions are properly defined to CICS, you can issue various operator-oriented CICS transactions (CEMT) through the z/OS MODIFY console command.

  The COF provides the additional CICS internal message traffic to detect automateable events. From these events, you can invoke complex automation procedures to handle most CICS subsystem availability and recovery issues for daily operations.

- Several CA OPS/MVS COF parameters enable you to WTO all transient data messages so that they can be displayed on a specific console, the system log, and in OPSLOG.

  Periodically, you may produce a special CICS heartbeat message, OPS3420O, to ensure that transaction activity is proceeding normally in CICS.

Some CICS Procedures You Can Automate

Using the combined features of the AOF and the COF, you can develop automation procedures to:

- Control the startup of CICS for normal, cold start, and emergency restart situations.
- Recover terminals and communication links that experienced communication failures.
- Monitor the logon and logoff activities of the users.
- Perform CICS journal archives when journals are full.
- Close groups of CICS files for batch processing at scheduled times.

Chapter 19: IMS Operation Facility

This section contains the following topics:

IMS IOF Overview

The IMS Operation Facility(IOF) is an optional feature of CA OPS/MVS. It allows CA OPS/MVS to operate one or more IMS Control Regions so as to seamlessly integrate them with the CA OPS/MVS operation of z/OS and JES.

The IOF is needed to operate the IMS because the IMS does not use the usual z/OS console interface to communicate with Operations. Instead, the IMS requires its own Master Terminal (MTO) that receives most (but not all) IMS-related messages. The IOF dynamically inserts an IMS Automated Operations Interface (AOI) exit to capture the IMS command and message traffic and also takes special care to identify and capture z/OS messages that are associated with the IMS.

CA OPS/MVS allows as many as 32 IMS Control Regions to be automated under the IOF. You can pre-define a total number of 16 IMS SVCs to the IOF for recognition.

IOF Installation Considerations

You may need to set CA OPS/MVS parameters that pertain to IMS control regions, for example, IMS1ID or IMS1DUPLICATE, during CA OPS/MVS installation. For information about these parameters, see the *Parameter Reference*. For information about installing the IOF, see the *Installation Guide*.

If IMS Type 2 messages and commands will be issued, two IMS modules included in the IMS RESLIB must be made available to CA OPS/MVS. For a detailed description of this requirement, and suggestions for several methods of implementation, see the *Installation Guide*.

IOF Installation Operations

Once installed, the IOF is only apparent as a set of extensions to the other facilities of CA OPS/MVS. For this reason, it is documented in the sections that describe the CA OPS/MVS facilities extended by the IOF.

Note: By accessing the CA OPS/MVS Identify IMS function (OPSVIEW Option 7.4), you can create the parameter cards necessary for the IMS Operation Facility. For information about OPSVIEW, see the *OPSVIEW User Guide*.

Interpreting Type 2 API Return and Reason Codes

Type 2 messages and commands utilize a specialized API to communicate with the IMSPLEX manager, and therefore are exposed to a new series of return and reason codes unique to conditions within the API.

Return and reason codes generated by this API have a distinctive format, where both return and reason codes are represented as:

X'*nnnnnnnn*'

For example, X'01000010'/X'00004004' is defined in the IBM documentation as CSLSRG00 could not be loaded, which indicates that the two required IMS modules are not available to CA OPS/MVS processing.

Note: The new series of type 2 return and reason codes are produced by IBM code, and documented in the IBM manual *IMS V10 System Programming API Reference* manual number SC18-9967-00, which is applicable to IMS version 10. Errors associated with API registration can be found under the section CSLSCREG Return and Reason Codes, and those with API dialog under CSLOMI Return and Reason Codes.

Issue Commands from a BMP Region

If a batch message processing (BMP) region is available, the IOF can use it to issue IMS commands and retrieve command responses. This ability is an alternative to CA OPS/MVS using the IMS WTOR method whenever it needs to issue an IMS command.

The IOF use of a BMP region for IMS commands has these advantages:

- The IOF can issue most IMS commands without waiting for the IMS WTOR.
- Command responses are more reliable and more efficient because command output is neither automatically routed to the consoles (as it would be if the commands were issued through the IMS WTOR) nor routed through the subsystem interface (SSI).

Perform these steps to take advantage of the CA OPS/MVS ability to use a BMP to issue IMS commands:

1. Set the IMS parameter AOIS= to a value other than N, which is the default value.

   Note: For a list of possible values, see the IMS installation manuals.

2. Authorize the BMP TRAN to Issue All Commands

   You must run the IMS Security Maintenance Utility (SMU) to specify that the BMP TRAN has authority to issue all commands.

3. Create a Batch BMP Started Task JCL

   Use the IMS PROCLIB member IMSBATCH, and make sure the RESLIB that it is using matches the RESLIB of the control region that you want to target. Add the CA OPS/MVS load module library to the STEPLIB concatenation.

   Set the CA OPS/MVS IMS*n*BMPSTC parameter to the member name of the BMP started task JCL.

4. Specify CA OPS/MVS Parameters

   To control the activation or deactivation of the BMP region that the IOF uses to issue commands, you need to set these CA OPS/MVS parameters:

   - IMSnBMPSTC
   - IMS*n*INITBMP
   - IMS*n*PSBNAME
   - IMS*n*TRANNAME
   - DEBUGBMP

   For more information about these parameters, see the *Parameter Reference*.

Note: You do not need to change any of your existing automation to take advantage of the ability of CA OPS/MVS to use a BMP in this way. As long as you have performed the steps above, you can continue issuing your IMS commands as you always have (for example, through the OPSCMD command processor or the ADDRESS OPER host environment) and CA OPS/MVS will use the BMP mechanism if it is available.

Chapter 20: NetView Operations Facility

This section contains the following topics:

About the NetView Operations Facility

The CA OPS/MVS NetView Operations Facility (NOF) component enables you to combine the CA OPS/MVS system automation capabilities with the network automation features of NetView.

The benefits of the NOF include:

- Two-way management for NetView alerts
- VTAM message handling
- An interface to the NetView STATMON (status monitoring) feature

The features available in the NOF enable NetView and CA OPS/MVS to function as a unit, with automation done where it logically belongs and with both CA OPS/MVS and NetView aware of the activities of each other.

NOF Alerts

A NetView alert provides information about system problems. Each alert consists of several pieces of information encoded (mostly) in two-byte hexadecimal strings called *code points*.

Information in these code points describes specific problem conditions. For example, code points can describe:

- An abend
- What probably caused a problem, for example, System Programmer Error
- The action to take to resolve a problem, for example, Restart the program
- The type of problem, for instance, Performance

Note: For more information about code points, see the IBM documentation.

Activate the NOF

When you installed the NOF, you copied the OPNOF REXX program into the NetView CLIST library. To test that you did this successfully, log on to NetView and issue the following command with no operands:

```
OPNOF
```

In response, you should see a message listing the valid parameters for the OPNOF program.

Note: For the steps to install the NOF, see the *Installation Guide*.

Parameters for the OPNOF Program

The OPNOF REXX program controls the functions that the NOF performs and how it executes those functions. To call the OPNOF program, use the following REXX code:

```
OPNOF   SET ALRTMLWTO ON|OFF
        SET ECHOVTAM WTO|MLWTO|NONE
        SET ECHOSTAT ON|OFF
        SET ALRTOPSGLV ON|OFF
        SET OPSID opsid1 (opsid2...opsidn)
        SHOW
        STAT|STATS
```

In calling OPNOF, you can specify any of these parameters:

SET ALRTMLWTO

Tells the NOF what to do when it receives a network alert. If you specify ON, the NOF sends the alert as a multi-line WTO message to the console, where your operators and CA OPS/MVS will see it. If you specify OFF, the NOF does not send alerts to the console.

If your main reason for sending alerts to the console is to allow CA OPS/MVS to see them, you may want to use the SET ALRTOPSGLV parameter instead.

SET ECHOVTAM

Tells the NOF what to do when it receives an unsolicited VTAM message. When NetView is active, unsolicited VTAM messages no longer flow to the console. SET ECHOVTAM gives you the option of echoing such messages to the console. If you use the PPOLOG option of VTAM (for example, F VTAM,PPOLOG=YES), the SET ECHOVTAM parameter also controls solicited command responses.

Possible subparameters for SET ECHOVTAM are:

- WTO-Issues single-line WTO messages for each line of the message.
- MLWTO-Causes the NOF to issue a multi-line WTO message to the console for the message.
- NONE-Tells the NOF not to echo the message to the console.

SET ECHOSTAT

Tells the NOF what to do when it receives a NetView status monitor update. The NetView status monitor receives such updates directly from VTAM over a proprietary interface and gives you the option of generating message CNM094I when one of these updates occurs.

If you specify SET ECHOSTAT ON, the NOF echoes these CNM094I messages to the console where CA OPS/MVS can see them. If you specify SET ECHOSTAT OFF, the NOF ignores status monitor updates.

SET ALRTOPSGLV

Tells the NOF whether to generate a CA OPS/MVS global variable called GLOBAL.OPNF.ALERT when it receives a NetView alert. This variable, generated if you specify SET ALRTOPSGLV ON, is generated in automatable format so that you can write global variable rules against it. This has the benefit of keeping alert information off the console if you only want to automate the response to the alert.

A sample rule called OPNFPALR processes the contents of the GLOBAL.OPNF.ALERT variable. If you specify SET ALRTOPSGLV OFF, the NOF generates no variable.

SET OPSID

Tells the NOF which CA OPS/MVS subsystems to send the global variable to. This command is meaningful only if you are sending alerts to CA OPS/MVS global variables. You can specify up to five CA OPS/MVS subsystems. For example, to generate global variables for alert in the CA OPS/MVS copy running under subsystem OPSS, you would specify the following command:

```
NOF SET OPSID OPSS
```

SHOW

Displays the values of the NOF parameters.

STAT or STATS

Displays the number of messages and alerts that the NOF has processed.

Because NOF parameters do not survive restarts of NetView, you should modify the initial CLIST that is invoked when NetView starts up so that the CLIST calls the sample NOF initialization routine. This routine, located in member OPNFINIT in the OPS.CCLXCLS0 data set, issues NOF SET commands for the default parameters.

To find your initial CLIST, look at the NCCFIC parameter in the DSIDMN member of the NetView parameter data set.

The NetView Alerts

In its most basic form, an alert is a control block called a Network Management Vector Transport (NMVT). The NMVT contains items called *vectors*, which may contain *subvectors*. Each vector or subvector represents part of the alert.

The NMVT control block also contains a few fields for text. Two important fields are:

HIERARCHY

This field is a network management construct that can be useful for both NetView-generated alerts and CA OPS/MVS generated alerts. The Hierarchy field describes the components that are involved in a failure. For example, a hierarchy can consist of two CICS regions and a DB2 region that jointly caused a single failure.

TEXT

This field enables the alert to include random text.

What Happens When You Generate an Alert

When NetView receives alerts, they flow into a NetView component called the hardware monitor, also known as the Network Problem Determination Analyzer (NPDA). NPDA both displays and manages alerts. From NPDA, you can automatically open records in the IBM INFO/Management product when problems occur.

To get into NPDA, issue the following command from any NetView command line:

```
NPDA
```

How the NOF Responds to NetView Alerts

In much the same way that CA OPS/MVS can send alerts into NetView, other system components (both hardware and software) also send alerts into NetView. This gives NetView access to an important systems management data stream that CA OPS/MVS can also tap into. Depending on your configuration, the alert stream can include DASD information, messages from NetView/6000, or even AS/400 messages. The next few sections explain how the NOF sees NetView alerts.

The NOF sees NetView alerts and responds:

- A function in NetView Version 2.2 and above allows the NetView automation table to see network alerts and drive automation based on those alerts.
- CA OPS/MVS supplies the entries in this automation table and the programs that do the automation. This enables CA OPS/MVS to capture all NetView alerts and forward those alerts to the CA OPS/MVS internal rules processor for resolution.
- The ALRTMLWTO and ALRTOPSGLBV parameters you specified when calling the OPNOF REXX program determine what the NOF does when it encounters an alert.
- Before sending the alerts to CA OPS/MVS, a REXX program called OPNFMSUR formats the information in the alert so that an automation procedure can make sense of it:
  - The OPNFMSUR program first translates the code points into English
  - Then places the result in a single global variable.

  You can then write a CA OPS/MVS global variable rule to process the alert.

Contents of GLOBAL.OPNF.ALERT

The GLOBAL.OPNF.ALERT global variable that the OPNFMSUR REXX program sends to CA OPS/MVS is specially formatted so that you can easily find the fields in the alert in which you are looking. The variable contents are formatted as follows:

alert-field-name=alert-field-value####

Alert-field-name is the name of one of the following CA NetView alert fields:

- ALERT_DESCRIPTION
- HIERARCHY_NUMBER
- ALERT_TEXT
- HIERARCHY_n
- ALERT_TYPE
- PROBABLE_CAUSE_NUMBER
- EVENT_CODE_NUMBER
- PROBABLE_CAUSE_n
- EVENT_CODE_TYPE
- RECOMMENDED_ACTION_NUMBER
- EVENT_CODE_n
- RECOMMENDED_ACTION_n

The *alert-field-value* is the value of the field for a particular alert. To process the alert, write a global variable rule modeled after the sample rule in member OPNFPALR of the OPS.CCLXRULB library. This sample rule echoes the contents of the alert to the console of the operator.

Alerts Generated from CA OPS/MVS

Generating alerts from CA OPS/MVS enables you to:

- Correlate network events and system events.
- Use NPDA to track problems.
- Notify NetView of events that NetView has no other way of seeing, such as OMEGAMON exceptions.

OPNFALRT REXX Function—Generate Alerts

CA OPS/MVS generates alerts through an OPS/REXX function called OPNFALRT. OPNFALRT accepts parameters, generates an NMVT, and passes the alert to NetView using the NetView program-to-program interface (PPI).

The OPNFALRT function is available in both rules and REXX programs. It is not available uner TSO/E REXX.

Note: If your data center runs CA products that use CA GSS, you should be aware that CA GSS Versions 2.5 and above have a NETVIEW function that operates exactly like the OPNFALRT function.

The REXX code for calling the OPNFALRT function has the following syntax:

```
rc = OPNFALRT(alerttype,
              alertdesc,
              probcause1;probcause2;probcause3,
              action1;action2;action3,
              hierarchy,
              alerttext)
```

alerttype

Specifies the type of alert to be generated.

alertdesc

Describes the condition for the alert.

probcause1, probcause2,...

Describes the probable cause for the problem.

action1, action2,...

Lets you specify actions to correct the problem.

hierarchy

Specifies all system resources relating to the problem.

alerttext

Lets you specify up to 136 bytes of arbitrary text.

Example: OPSFALTR function

This sample code demonstrates the use of OPNFALRT:

```
/* rexx */
rc = OPNFALRT('PERF',,         /* Performance alert */
              'X1111',,        /* Code point 1111 for description*/
              'IODEVICE',,     /* Probable cause is an IODEVICE */
              'X2111;VARYOFF',,/* First action is code point 2111*/
```

```
                                /* second action is VARY OFFLINE */
              'IMSPROD IMS DASD721 DASD',,
                                /* hierarchy includes IMS and DASD*/
              'Alert detected and processed by CA OPS/MVS')
                                 /* text message */
if word(rc,1) □ 0 then,
do
  say 'Return code from OPNFALRT was' word(rc,1)
  say subword(rc,2)
end
```

Alert Type Parameter

This parameter specifies the type of alert to be generated. Specify one of the following *alerttype* values:

Value	Generates an alert reporting...
AVAL	A loss of system availability
BYPS	An alert bypassed condition
CUST	A condition caused by an end user
DLRC	A delayed response condition
IMPD	Something that will cause immediate impact on your system
IMRE	Intensive mode recording
INST	The installation of a product or component
INTV	An intervention required condition
NTFY	A system notification
PAFF	A permanently affected resource
PERF	A condition that will affect system performance
PERM	A permanent condition
PROC	An operator procedure error
SCUR	A security violation
TEMP	A temporary condition
REDL	The loss of a redundant system
UNKN	This alert concerns a condition that is of an unknown type or does not fall into one of the above categories

Alert Description Parameter

This parameter is a code point describing the condition for the alert. Code points in the OPNFALRT function can be one of two things:

- A five-character string, starting with the letter X. The other four characters are the two-byte hexadecimal code point for the description. You can find valid code points for the description in the IBM documentation.
- A word describing the condition. You can find the supported words in the *SNA Format Reference Guide.*

The alert description parameter is required and you can specify it only once.

You can enter one of the following words instead of a code point:

Word	Means that an alert concerns...
ABEND	A software program that abended
INCOROUT	A program providing incorrect output
PERFDEGRADED	A situation that is degrading performance
OUTOFRESOURC	A situation involving a depleted resource
FILEREORG	A situation involving a file that needs to be reorganized
OPERERR	An operator error
RESNOTACT	An error caused by an inactive resource
CONFIGERR	An error caused by a configuration error
OPNOTIFY	An operator notification
SECURITYEVNT	A security event

Probable Cause Parameter

You can use the *probcause* parameter to enter one to three code points describing a probable cause for the problem. You must specify at least one probable cause parameter.

The code points for *probcause* are similar to the ones described for the *alertdesc* parameter. If you specify more than one code point, separate the code points with a semicolon (;). You can mix and match English-language format with hexadecimal (X*nnnn*) format.

Words you can specify in place of code points are:

Word	Means that the probable cause of the error is...
PROCESSOR	The processor
SOFTWARE	Software
APPLPROG	An application program
IOACCMETH	An I/O access method
COMM	Communications problems
IODEVICE	An I/O device
PRINTER	A printer
DASD	A DASD
CONSOLE	A console

If you need to specify a probable cause different from those listed above, use the X*nnnn* format. The *SNA Format Reference Guide* documents probable cause code points. If you specify more than one probable cause parameter, specify them in decreasing order of probability. For example, you might specify the probable cause of a CICS failure SOFTWARE;IOERROR when software is more likely to cause the problem than an I/O error.

Action Parameter

Use the *action* parameter to enter one to three possible actions to correct the problem. You must specify at least one alert action parameter.

The code points for *action* are similar to those for the *alertdesc* parameter. If you specify more than one action parameter, separate them with a semi-colon (;). You can mix and match the *Xnnnn* format with English word format. Valid words for action code points are as follows:

Word	Specifies this action...
PROBDETERMIN	Perform problem determination procedures
TAKEADUMP	Generate a dump
DUMP	Generate a dump
PERFPROBREC	Perform problem recovery procedures
PROBRCVY	Perform problem recovery procedures
REFERGUIDE	See the *SNA Formats Reference Guide*
VARYOFF	Vary the failing component offline
CONTACTREP	Contact a service representative
RETRY	Retry the failing procedure
RESTARTJOB	Restart the failing job

If you need to specify an action different from those listed above, use the X*nnnn* format. The *SNA Formats Reference Guide* describes the action code points. If you specify more then one action, put them in the order in which you want the operator or network manager to try to implement them.

Hierarchy Parameter

Use the *hierarchy* parameter to specify all the system resources relating to the problem. This enables NPDA to draw a picture of these resources, and also enables you to filter alerts based on the resources. Specify the hierarchy parameter as a character string containing 0 to 60 bytes, which can contain up to five 12-byte entries. Each 12-byte entry consists of:

- An eight-byte resource name. This is arbitrary, and can be something like IMSPROD.
- A four-byte resource type. This is also arbitrary, but can be something like IMS or INIT.

Suppose that you want to generate an alert because of an IMS failure. You might specify the hierarchy parameter as follows:

```
IMSPROD IMS
```

Note the spacing of the blanks in this string. In any given 12-byte hierarchy item, the first eight bytes are the resource name and the next four bytes are the resource type. This structure is then copied up to five times to make the complete hierarchy parameter.

Now suppose that the IMS failure in the previous example resulted from an I/O error. You can specify the hierarchy parameter as follows:

```
IMSPROD IMSDASD721 DASD
```

The above parameter specifies that the IMS outage resulted from a problem with the resource name DASD721 of type DASD.

The hierarchy parameter is optional. If you omit it, the OPNFALRT function specifies it for you, but you must include a comma if you want to specify the alert text parameter.

Alert Text Parameter

The *alerttext* parameter allows you to specify up to 136 bytes of arbitrary text. This text will be available on the NetView NPDA alert screens. You might use this parameter to include the message that triggered the alert, or to clarify the conditions under which you issued the alert.

The alert text parameter is optional. If you do not specify it, the OPNFALRT function includes the phrase ALERT GENERATED BY CA OPS/MVS.

OPNFALRT Return Code Format

Return codes from OPNFALRT have a slightly different format than return codes from other OPS/REXX functions. Executing OPNFALRT generates both a return code and a description of what the return code means, and OPNFALRT places nothing in the OPS/REXX external data queue.

OPNFALRT return codes have this format:

- The first word is a numeric value.
- The second word is a message identifier.
- The rest of the return code describes the error.

The OPS/REXX code shown above shows how to process return codes from OPNFALRT.

More information:

OPNFALRT REXX Function—Generate Alerts (see page 602)

OPNFALRT Messages and Return Codes

This section provides explanations of OPNFALRT messages, and the return codes that correspond to them.

OPNF001E

Could not find CMNETV in LPA

Reason:

You tried to issue the OPNFALRT from a CA OPS/MVS rule, but the CA OPS/MVS NetView interface module CNMNETV is not in the system LPA.

Return code 4-CA OPS/MVS could not find the CNMNETV interface module.

Action:

Do one of the following:

- Copy CNMNETV and re-IPL your system.
- Use a product such as the CA SYSVIEW to dynamically load CNMNETV into LPA.
- Issue the OPNFALRT function only from OPS/REXX programs.

OPNF002E

OPNF002E Could not load CNMNETV

Reason:

You tried to issue OPNFALRT from an OPS/REXX program, but CA OPS/MVS could not load its CNMNETV module. The CNMNETV module must be available in the system LPA, the LINKLIST, or the STEPLIB concatenation for the program issuing the OPNFALRT function.

Return code 4-CA OPS/MVS could not find the CNMNETV interface module.

Action:

Make CNMNETV available to that program.

OPNF003E

Specified description not in table

Reason:

The English-language value you used for the alert description parameter is not a valid description. Therefore, OPNFALRT rejected your alert.

Return code 8-The call to OPNFALRT had an invalid description field. Message OPNF003E or OPNF004E accompanies this return code.

Action:

Specify a valid description value and retry the alert.

OPNF004E

Specified code point is invalid

Reason:

The code point value you specified for the alert description parameter was not in the correct format, so OPNFALRT rejected your alert. A code point starts with the letter X and is followed by exactly four hexadecimal characters.

Return code 8-The call to OPNFALRT had an invalid description field. Message OPNF003E or OPNF004E accompanies this return code.

Action:

Specify a valid description value and retry the alert.

OPNF005E

More than three probable causes specified

Reason:

You tried to specify more than three (the maximum allowed) probable cause parameters. Therefore, OPNFALRT rejected your alert.

Return code 12-The call to OPNFALRT had an invalid probable cause parameter. Message OPNF005E, OPNF006E, or OPNF007E accompanies this return code.

Action:

Specify a valid number of parameters, and be sure to separate them with semicolons.

OPNF006E

Error in PC *n* - PC was not in table

Reason:

You tried to specify a probable cause in English format, but the parameter value you used is not valid so OPNFALRT rejected your alert. The *n* value indicates the position of the invalid parameter in a string of parameters.

Return code 12-The call to OPNFALRT had an invalid probable cause parameter. Message OPNF005E, OPNF006E, or OPNF007E accompanies this return code.

Action:

Substitute a valid parameter value for the one in error and retry the alert.

OPNF007E

Error in PC *n* - PC code point invalid

Reason:

You tried to specify a probable cause parameter in hexadecimal format, but the parameter value you used is not valid so OPNFALRT rejected your alert. The *n* value indicates the position of the invalid parameter in a string of parameters.

Return code 12-The call to OPNFALRT had an invalid probable cause parameter. Message OPNF005E, OPNF006E, or OPNF007E accompanies this return code.

Action:

Substitute a valid parameter value for the one in error and retry the alert.

OPNF008E

More than three action parameters specified

Reason:

You tried to specify too many action parameters, so OPNFALRT rejected your alert.

Return code 16-The call to OPNFALRT had an invalid action parameter. Message OPNF008E, OPNF009E, or OPNF010E accompanies this return code.

Action:

Retry the alert, specifying no more than three action parameters. Be sure to separate the parameters with semicolons.

OPNF009E

Error in action code *n* - action was not in table

Reason:

The English-format parameter indicated by *n* specifies an invalid action. Therefore, OPNFALRT rejected your alert.

Return code 16-The call to OPNFALRT had an invalid action parameter. Message OPNF008E, OPNF009E, or OPNF010E accompanies this return code.

Action:

Retry the alert, specifying a valid action.

OPNF010E

Error in action *n* - action code point invalid

Reason:

The code point you specified for the parameter indicated by *n* is an invalid code point.

Return code 16-The call to OPNFALRT had an invalid action parameter. Message OPNF008E, OPNF009E, or OPNF010E accompanies this return code.

Action:

Replace the invalid code point with one that starts with the letter X followed by four hexadecimal characters. Then, retry the alert.

OPNF011E

Hierarchy format invalid

Reason:

The format of the hierarchy parameter is invalid. This parameter is free format, but the value you specify for it must contain a number of characters that is evenly divisible by 12.

Return code 20-The call to OPNFALRT had an invalid hierarchy parameter. Message OPNF011E accompanies this return code.

Action:

Correct the hierarchy parameter value and retry the alert.

OPNF012E

Text format invalid

Reason:

The text parameter value contains too many characters.

Return code 24-The call to OPNFALRT had an invalid alert text parameter. Message OPNF012E accompanies this return code.

Action:

Specify a parameter value that has 136 or fewer characters and retry the alert.

OPNF013E

Error calling OPNFGENR RC=*xxxx*

Reason:

The OPNFALRT function encountered an internal error.

Return code 28-The call to OPNFALRT encountered an error. Message OPNF013E describes this error.

Action:

Contact CA Technical Support. If you can reproduce the error, convert the return code to hexadecimal format using the C2X function of REXX.

OPNF014E

NetView PPI is not active

Reason:

The NetView program-to-program interface was not active when you invoked the OPNFALRT function, probably because the NetView subsystem address space was not active.

Return code 32-The call to OPNFALRT encountered an error. Message OPNF014E describes this error.

Action:

Start the NetView subsystem address space. If the error persists, call CA Technical Support.

OPNF015E

Error in NetView PPI, RC=*xxxx*

Reason:

The NetView program-to-program interface returned an unexpected return code to OPNFALRT. The *xxxx* is the return code from the NetView PPI.

Return code 36-The call to OPNFALRT encountered an error. Message OPNF015E describes this error.

Action:

Contact CA Technical Support.

OPNF016E

CNMCALRT is not active in NetView

Reason:

The CNMCALRT task is not active in the NetView address space. This task is required if alerts will be processed over the NetView program-to-program interface.

Return code 40-The call to OPNFALRT encountered an error. Message OPNF016E describes this error.

Return code 44-The call to OPNFALRT encountered an error.

Action:

Start the CNMCALRT task using the following NetView command:

```
START TASK=CNMCALRT
```

This task normally starts automatically when NetView initializes. Contact your NetView system programmer if this problem persists. If the CNMCALRT task is active and you receive message OPNF016E, call CA Customer Support.

OPNF017E

OPNFALRT not executing in primary mode

Reason:

An internal error occurred while the alert was being processed.

Return code 48-The call to OPNFALRT encountered an error.

Action:

Contact CA Customer Support.

OPNF019E

Less than three or more than six parameters

Reason:

You specified either too many or too few parameters to OPNFALRT.

Return code 52-The call to OPNFALRT encountered an error. Message OPNF019E describes this error.

Return code 60-The call to OPNFALRT encountered an error. Message OPNF021E describes this error.

Action:

Correct the call to OPNFALRT, specifying a valid number of parameters. Then, retry the alert.

OPNF021E

TYPE parameter invalid

Reason:

The alert type parameter is not valid, either because it does not contain exactly four characters or because it does not specify one of the alert type values.

Return code 56-The call to OPNFALRT encountered an error.

Action:

Specify a correct type parameter value and retry the alert.

Issuing NetView Commands

NetView supports a console interface, so you can issue NetView commands and receive the command responses. To issue a NetView command, you need to know the NetView system recognition character and to establish NetView autotasks for each console from which you will issue NetView commands.

Establish NetView Autotasks

NetView requires an autotask for every console that it might receive a command from. When you issue a command through the ADDRESS OPER host environment in a rule, CA OPS/MVS usually uses the console ID of the master console. Therefore, you should at least establish an autotask for the z/OS master console. Doing this enables you to issue NetView commands from CA OPS/MVS rules, but not to retrieve the responses.

To establish an autotask for the master console, issue this NetView command:

```
AUTOTASK OPID=operator,CONSOLE=mstrconsole
```

operator

Specifies a NetView operator ID defined in the DSIOPF member of the NetView parameter data set.

mstrconsole

Specifies the name of your master console.

Retrieve Responses to NetView Commands

If you want OPS/REXX programs to issue NetView commands and receive command responses, these programs must run in an environment where operator command responses are returned. When you issue a command, CA OPS/MVS chooses an available subsystem console from its pool of subsystem consoles. Because you do not know which console CA OPS/MVS will select, you need to define *all* CA OPS/MVS subsystem consoles to NetView using the autotask command.

The DEFNVCON program in the CA OPS/MVS OPS.CCLXSAMP library is a sample OPS/REXX program that issues the NetView autotask command for each CA OPS/MVS subsystem console.

To define autotasks to retrieve responses to NetView commands

1. Use the DEFNVCON sample program to define the consoles.

   This guarantees that the current set of CA OPS/MVS subsystem consoles are always defined to NetView, even if your subsystem consoles change from one CA OPS/MVS startup to another startup.

2. Before running DEFNVCON, do the following tasks:
   - Change the assignment statements in the beginning of the DEFNVCON sample to reflect your naming standards for NetView user IDs.
   - Define the maximum number of CA OPS/MVS subsystem consoles you want to define to NetView.

   The default operator ID prefix is OPSMVS, and DEFNVCON will define all CA OPS/MVS subsystem consoles to NetView.

The autotasks are defined for all of your consoles.

For more information, see the comments in DEFNVCON.

Find NetView System Recognition Character

Once you have defined NetView autotasks for all of your consoles, you can issue NetView commands from z/OS consoles or from CA OPS/MVS using the OPSCMD function in OPS/REXX or the ADDRESS OPER host environment. However, you also need to know the subsystem recognition character to issue commands.

To find the NetView system recognition character

1. The OPSNETV function of OPS/REXX returns the names and subsystem recognition characters of all NetView copies. This function has the following syntax:

   ```
   rc= OPSNETV('I','SRC')
   ```

   This function returns a character string composed of one or more three-word substrings called *triplets*. In each triplet:
   - The first word is the subsystem recognition character.
   - The second word is the name of the job running NetView.
   - The third word is the name of the NetView subsystem address space.
2. Review the DEFNVCON sample program and the OPNFFNET and OPNFSEC sample rules.

   They demonstrate the use of the OPSNETV function.

Example:Find the Recognition Character

Suppose that the OPSNETV function returns the following string:

```
% NETV230 NETVSSI # NV240 NV240SSI
```

This string indicates that you have two versions of NetView running. The first runs in address space NETV230; its address space is NETVSSI and its subsystem recognition character is %. The other NetView version runs in address space NV240 with a subsystem recognition character of #.

Chapter 21: Using the Automation Measurement Environment

This section contains the following topics:

Overview of AME

The Automation Measurement Environment(AME) collects, records, and reports on the system events that CA OPS/MVS reacts to and the actions that it takes to address those events.

CA OPS/MVS continuously records statistical data. Much of this data can be obtained online as well as in OPS/REXX programs using product facilities such as:

- OPSVIEW option 4.1 and its suboptions
- OPSVIEW option 7.2 (the Automation Analyzer)
- The OPSPRM OPS/REXX function or the OPSPARM TSO command
- The ADDRESS OPSCTL OSF host command environment in OPS/REXX
- The ADDRESS AOF host command environment in OPS/REXX

You can also request that CA OPS/MVS write SMF records. The SMF records will go to SMF data sets or the MVS System Logger (LOGR) depending on your system definitions. CA OPS/MVS produces two types of SMF records:

- Summary-type records that are written at product shutdown
- Event-type records that are written after certain events are completed

Required Software

To run the AME, you must have CCS for z/OS installed. For more information on the CCS for z/OS component that is required to run AME, see the appendix “CCS for z/OS Component Requirements” in the *Installation Guide*.

Note: This software is automatically shipped with CA OPS/MVS.

Advantages of the AME

The AME provides these advantages:

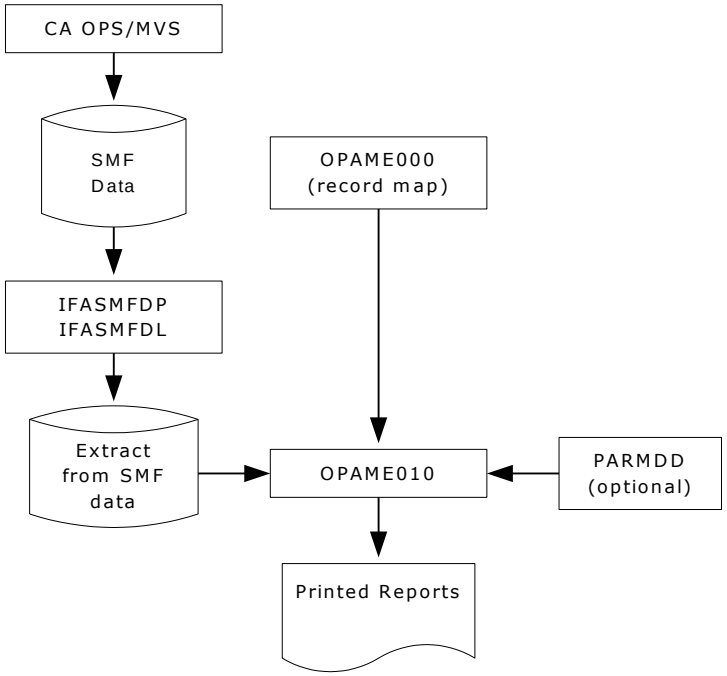
- It produces a detailed statistics report, the Automation Statistics Report, to help you monitor your automation functions.
- It lets you print and retain historical automation information.
- It lets you use the PARM='PARMDD=*ddname*' to tailor the report so that it includes only the information you need.

Data Flow of the AME

The OPAME010 program scans data streams for SMF records. It then summarizes the data and prints it in an Automation Statistics Report.

The following diagram shows the flow of the data that CA OPS/MVS collects:

Equation 2: Flow of data collected by CA OPS/MVS



Here is a description of some of the important items shown in the previous diagram:

- CA OPS/MVS

  Your copy of CA OPS/MVS.

- SMF data

  The SMF data sets or System Logger logstream of the operating system, which are the possible destination for the SMF records that CA OPS/MVS collects. The destination is dependent on your system settings.

- IFASMFDP or IFASMFDL
  - IFASMFDP for SMF dump data sets

    If you choose to send SMF records to the data set of the operating system, IFASMFDP is the IBM utility used to extract the exact data you want. For a sample of the JCL, see OPS.OPS.CCLXCNTL(OPAMESMF).
  - IFASMFDL for MVS System Logger logstream

    If you choose to send SMF records to the MVS System Logger, IFASMFDL is the IBM utility used to extract the exact data you want. For a sample of the JCL, see OPS.CCLXCNTL (OPAMESML).
- OPAME010

  The program that summarizes the data and produces an Automation Statistics Report. It will accept the data that IFASMFDP extracted from the SMF data set of the operating system or the data that IFASMFDL extracted from the logstream.
- OPAME000

  The record map for SMF records. This is provided in OPS.CCLXCNTL(OPAME000).
- PARMDD

  A JCL subparameter that indicates the ddname from which OPAME010 reads additional subparameters. See OPS.CCLXCNTL(OPAMERPT).

Types of AME Reports

The AME reports on statistical summaries and event-type data that is recorded while CA OPS/MVS is active. CA OPS/MVS currently records only summary information; however, the ability to record during intervals will be added to the product in a future release.

Following is a list of the statistical segments that the AME reports on:

AOFEVENT

Reports AOF event statistics

OSFEVENT

Reports OSF event statistics

OSFTERM

Reports OSF server termination event statistics

IMS

Reports IMS BMP event statistics

GENERAL

Reports general summary statistics

MESSAGE

Reports WTO/WTOR message summary statistics

PROCBLK

Reports virtual storage Process Block summary statistics

EVNTEXIT

Reports exit call frequency summary statistics

AOFSUMM

Reports automated Operations Facility (AOF) summary statistics

OSFSUMM

Reports operator Server Facility (OSF) summary statistics

COMMAND

Reports command processing summary statistics

PERMVAR

Reports permanent global variable summary statistics

TEMPVAR

Reports temporary global variable summary statistics

OPSVALUE

Reports OPSVALUE processing summary statistics

SQL

Reports SQL summary statistics

EPI

Reports EPI summary statistics

Define Destinations and Intervals for SMFLOG Records

Four CA OPS/MVS parameters determine how CA OPS/MVS collects and records your SMF records. These parameters are:

- SMFRECORDING

  Controls whether CA OPS/MVS writes SMF records after the SMFRECORDNUMBER is set to a valid number.

- SMFRECORDNUMBER

  Sets the SMF record number used by CA OPS/MVS.

  Use this parameter to indicate whether you want CA OPS/MVS to generate SMF records. By default, the SMFRECORDNUMBER parameter is set to 0 (zero) and no SMF records are generated.

- SMFRULEDISABLE

  Controls whether CA OPS/MVS will write an SMF record when a rule, rule set, or both are disabled.

- OSFTRANSSMFREC

  Controls whether CA OPS/MVS creates SMF records after each server transaction.

For more information about these parameters, see the *Parameter Reference*.

Define the Content of the Automation Statistics Report

An EXEC statement in your JCL identifies the program that creates and prints your Automation Statistics Report; this program is OPAME010. The AME lets you tailor your report by specifying variable information for the PARM parameter of the JCL on this EXEC statement. The PARM parameter passes this information to OPAME010 when it executes. You can direct CA OPS/MVS to read parameters from a data set you specify in the PARM parameter itself. For example:

```
// EXEC PGM=OPAME010,PARM='PARMDD=ddname'
```

You can use the subparameters specified in the *ddname* to specify the variable information. These subparameters are not CA OPS/MVS parameters. The sample batch JCL to submit an Automation Statistics Report is in member OPAMERPT of the OPS.CCLXCNTL data set.

Subparameters Specified In the PARMDD File

The PARMDD itself is specified in the PARM field shown in the example above. If PARMDD is not specified, the defaults are taken for the report.

You can specify the following subparameters on the PARMDD file:

REPORTDD

Indicates the ddname to which the OPAME010 writes.

Syntax:

REPORTDD={SYSPRINT|*ddname*}

SYSPRINT

OPAME010 writes to SYSPRINT.

ddname

OPAME010 writes to the ddname you specify in place of the variable *ddname.*

Default: REPORTDD=SYSPRINT

LINECNT

Indicates the number of lines per page of the report.

Syntax:

LINECNT=*nn*

nn

Indicates the number of lines you want per page; if you specify LINECNT=0, CA OPS/MVS issues the titles only once.

Default: LINECNT=60

FROMDATE

Filters out data older than a date you specify.

Syntax:

FROMDATE={*yyyy.ddd*|*yyyy/mm/dd*|*nnn*}

yyyy.ddd

Indicates the cutoff date for data to appear on the report. Data older than the date you specify instead of *yyyy.ddd* will not appear.

yyyy/mm/dd

Indicates the cutoff date for data to appear on the report. Data older than the date you specify instead of *yyyy*/*mm*/*dd* will not appear.

nnn

Specifies a value to indicate the data cutoff date. For example, 0 is today, -1 is yesterday, and so on.

Default: Data from any date not greater than the value for TODATE appears. If you do not specify TODATE, data from all dates appears.

TODATE

Filters out data newer than a date you specify.

Syntax:

TODATE={*yyyy.ddd*|*yyyy/mm*/dd|*nnn*}

yyyy.ddd

Indicates the cutoff date for data to appear on the report. Data newer than the date you specify instead of yyyy.ddd will not appear.

yyyy/mm/dd

Indicates the cutoff date for data to appear on the report. Data newer than the date you specify instead of yyyy/mm/dd will not appear.

nnn

In place of nnn, specify a value to indicate the data cutoff date. For example, 0 is today, -1 is yesterday, and so on.

Default: Data from any date not less than FROMDATE appears on the report.

FROMTIME

Filters out data older than a time you specify.

Syntax:

FROMTIME=*(hh:mm)*

hh:mm

Indicates the cutoff time for data to appear on the report. Data older than the time you specify instead of *hh:mm* will not appear.

Default: FROMTIME=(00:00)

TOTIME

Filters out data newer than a time you specify.

Syntax:

TOTIME=*(hh:mm)*

hh:mm

Indicates the cutoff time for data to appear on the report. Data newer than the time you specify instead of *hh:mm* will not appear.

Default: TOTIME=(24:00)

RECMEGS

Specifies the SMF record accumulation buffer size in megabytes. The buffer is allocated from extended private virtual storage.

Syntax:

RECMEGS=*nnn*

nnn

Indicates the number of megabytes allocated for the SMF record accumulation buffer. You can specify a number from 1 to 100.

Default: RECMEGS=3

RECTYPE

Because the same data is collected in both event and summary records, you must tell the OPAME010 program to process only event records or only summary records for a given pass. Use the RECTYPE subparameter to indicate which type of records you want OPAME010 to process.

Syntax:

```
RECTYPE={EVENT|SUMMARY}
```

EVENT

Indicates you want OPAME010 to process only event records.

SUMMARY

Indicates you want OPAME010 to process only summary records.

Default: RECTYPE=EVENT

SMFID

Filters out data so that only those SMF records from systems you specify are included in the report.

Syntax:

SMFID={ALL|*(smfids)*}

ALL

Includes SMF records from all systems in the report.

smfids

Indicates that you do not want data from SMF records of all systems to be included in the report. SMF records from only those systems you specify will be included.

Default: SMFID=ALL

SMFTYPE

Specifies the SMF record type of the CA OPS/MVS SMF records to distinguish them from other records sent to the SMF data set. This subparameter should always match the SMFRECORDNUMBER parameter.

Syntax:

SMFTYPE=*nnn*

nnn

Indicates the number you want to assign to the SMF record. You can specify a number from 128 to 255.

Default: There is no default for this subparameter.

SSNM

Filters out data so that only the SMF records from subsystems that you specify are included in the report.

Syntax:

SSNM={ALL|(*ssnms*)}

ALL

Includes SMF records from all subsystems in the report.

ssnms

Indicates that you do not want data from SMF records of all subsystems to be included in the report. SMF records from only those subsystems you specify will be included.

Default: SSNM=ALL

WIDTH

Indicates the maximum width of the Automation Statistics Report.

Syntax:

WIDTH={80|132}

80

Prints the report with a width of 80 characters.

132

Reports the report with a width of 132 characters.

Default: WIDTH=132

OPSSTATS

Indicates whether you want CA OPS/MVS to create an Automation Statistics Report.

Syntax:

OPSSTATS={NO|(YES,[*values*])}

NO

Specifies that CA OPS/MVS will not create a report of statistical data.

YES

Specifies that CA OPS/MVS creates a report of statistical data. If you specify OPSSTATS=YES, there are additional values that you can specify; for a list of these values, see the table on the following page.

Note: These additional values are optional and will default if not specified.

Default: OPSSTATS=NO

Values You Can Specify for the OPSSTATS Subparameter

Here is a list of the values that you can specify for the OPSSTATS subparameter:

FROMDATE

This date overrides the global FROMDATE you specify.

Syntax:

FROMDATE={*yyyy.ddd*|*yyyy/mm/dd*|*nnn*}

TODATE

This date overrides the global TODATE you specify.

Syntax:

TODATE={*yyyy.ddd*|*yyyy/mm/dd*|*nnn*}

FROMTIME

This time overrides the global FROMTIME you specify.

Syntax:

FROMTIME=(*hh:mm*)

TOTIME

This time overrides the global TOTIME you specify.

Syntax:

TOTIME=(*hh:mm*)

SEGMENTS

If you specify SEGMENTS=ALL, CA OPS/MVS includes all statistics segments in the report. If you do not want all statistics segments to be included in the report, indicate those you want. For example, if you specify SEGMENTS=(SUMMARY,AOFEVENT, your report includes only summary and AOF event statistics.

Syntax:

SEGMENTS={ALL|(*segments*)}

JCL PARM Parameters

To simplify the typical batch JCL, all of these subparameters have defaults. Therefore, you may need to specify only the name of the report.

Examples: JCLPARM parameters

The following example shows JCL that requests an Automation Statistics Report that includes all statistics segments. This is true because the SEGMENTS value, which defaults to ALL, is not specified in the JCL.

```
//STEP1    EXEC PGM=OPAME010,PARM='PARMDD=PARMFILE'
//OPSSMF   DD DISP=SHR,DSN=ops.smflog.dataset.name
//OPSSRM   DD DISP=SHR,DSN=ops.cclxcntl(OPAME000)
//SYSPRINT DD SYSOUT=*
//PARMFILE DD *
OPSSTATS=YES,SMFTYPE=215
```

The following example shows JCL that requests an Automation Statistics Report that includes only event data for OSF servers and disabled AOF rules. This JCL defines the report further by specifying that only data between the hours of 8:00 a.m. and 5:00 p.m. should be included and that the report should print with a width of 80 characters.

```
//STEP1    EXEC PGM=OPAME010,PARM='PARMDD=PARMFILE'
//OPSSMF   DD DISP=SHR,DSN=ops.smflog.dataset.name
//SYSPRINT DD SYSOUT=*
//PARMFILE DD *
WIDTH=80,OPSSTATS=YES,
SEGMENTS=(OSFEVENT,AOFEVENT),FROMTIME=(8:00),
TOTIME=(17:00)'
```

The OPS.CCLXSAMP (OPAMEPRM) provides an example of how to build a REXX EXEC for the parameters, and then have them go to a data set.

Generate the Summary Section

The statistics reported in the Summary Section of the Automation Statistics Report are accumulated throughout the execution of CA OPS/MVS. When CA OPS/MVS is terminated, these statistics are totaled and written to an SMF record. The OPSSMF OPS/REXX function can also be used to write the cumulative summary records on demand, instead of upon product termination. One reason to do this is when the SMF address space is terminated prior to CA OPS/MVS terminating. For more information on the OPSSMF function, see the *Command and Function Reference*. The Summary Section reports the data in this SMF record.

To generate the Summary Section, specify SEGMENTS=(ALL) and RECTYPE=(SUMMARY).

Example: Summary Section

Here is an example of a Summary Section:

```
                Automation Measurement Environment
Statistics For SMFID S034     First Record Start: 2009/08/07 08:04:03
SSNM OPSS                        Last Record End:            08:04:31
                   GENERAL: General Summary Statistics
1>                   General Summary Statistics
    Approximate Product Start Time. . . . . . . . .2009/08/07 07:40:08
    Time Initialization Completed . . . . . . . . .2009/08/07 07:40:29
    Time Shutdown Started . . . . . . . . . . . .  . 2009/08/07 08:04:00
2>                 MESSAGE: Message Summary Statistics
    NORMAL   Messages Processed . . . . . . . . . . . . . . .  734
    SUPPRESS Messages Processed . . . . . . . . . . . . . . . 18
    DISPLAY  Messages Processed . . . . . . . . . . . . . . .  0
    DELETE   Messages Processed . . . . . . . . . . . . . . .  0
    NOOPSLOG Messages Processed . . . . . . . . . . . . . .  448
    Generic Dataset Interface Msgs Processed. . . . . . . . .  0
    Maximum NOOPSLOG MSG Rules Active . . . . . . . . . . . .  5
    Messages bypassed due to AOFMPFBYPASS . . . . . . . . . .  0
    Messages bypassed due to BYPASSCMDECHO. . . . . . . . . . 0
    Internal product messages bypassed. . . . . . . . . . . . 69
    Messages bypassed due to JES3 processing. . . . . . . . .  0
    Duplicate IMS messages bypassed . . . . . . . . . . . . .  0
    Messages bypassed miscellaneous reasons . . . . . . . . .  0
    Messages bypassed due to AOF inactive . . . . . . . . . .100
3>               PROCBLK: Process Block Summary Statistics
    High Water Mark Block Usage Time. . .  2009/08/07   11:47:41
    Last Block Allocation Failure Time. . . . . . . . .   <none>
    Total Number of Blocks Used . . . . . . . . . . . .  2,860
    Number of Block Allocation Failures . . . . . . . . .  0
    High Water Mark Block Usage Count . . . . . . . . . .  7
    Actual Allocated Process Block Count. . . . . . . . . . 27
                  REXX Workspace Summary Statistics
    High-Used RXWS Water Mark Time. . . . . 2009/08/07  11:41:13
    RXWS Allocated Space (AOFSIZE parm) . . . . . . . . .  .256K
    High-Used RXWS Water Mark . . . . . . . . . . . . . . 41,472
    High-Used RXWS Address. . . . . . . . . . . . . . . 0635A000
    High-Used RXWS Water Mark Program . . . . . . . INT.OPS50060
    High-Used RXWS Water Mark Rule Name . . . . . . INT.OPS50060
                AOF Message Queue Summary Statistics
    Max pending GLV events (GLVPENDINGMAX). . . . . . . . .  100
    High-used GLV pending message queue . . . . . . . . . . ...0
    Maximum AOF EDQ entries (AOFMAXQUEUE) . . . . . . . .  3,001
    High-used AOF External Data Queue . . . . . . . . . . .  6
```

```
4>              EVNTEXIT: Event Exit Summary Statistics
                 Exit Call Summary Statistics
    IATUX18 Exit Calls. . . . . .  0
    IATUX31 Exit Calls. . . . . .  0
    Job Selection Calls . . . . .  0
    Free/Deallocate Calls . . . .  0
    End Of Memory Exit Calls. . .  1
    WTO Exit Calls. . . . . .921
    Command Exit Calls. . . . . 22
    Job Termination Calls . . . .  3
    Job Re-enqueue Calls. . . . .  0
    DOM (SVC 87) Calls. . . . . 10
    Open/Close/Checkpoint/Restart Calls . . . . . 12
    Step Initiation calls . . . .  0
    Failing SVC 34 Calls. . . . .  0
    Write To Log Calls. . . . . 989
    SUBSYS Keyword Converter Exit Calls . . . . .  6
    Allocation Grouping of SUBSYS Calls . . . . .  3
    Alternate End Of Task Calls . . . . . 209
    Other Subsystem Exit Calls. . . . . .  0
    IMS AOI Exit Message Calls. . . . . .  0
    IMS AOI Exit Command Calls. . . . . .  0
    SVC 95 function routine Exit Calls. . . . . .  4
    SVC 95 function routine bypassed. . . . . .  0
5>                  AOFSUMM: AOF Summary Statistics
                    AOF Event Summary Statistics
    AOF Rules Bypassed (LIMIT). . . . . .  0
    AOF Events Bypassed . . . . . .169
    AOF COMMAND Events. . . . . 22
    AOF Rule DISABLE Events . . . . . 26
    AOF DOM Events. . . . . .0
    AOF Rule ENABLE Events. . . . . 26
    AOF EOM Events. . . . . .1
    AOF GLV Events. . . . . .0
    AOF MESSAGE Events. . . . . 752
    AOF OMEGAMON Events . . . . . .  0
    AOF REQUEST Events. . . . . .  1
    AOF SECURITY Events . . . . . 268
    AOF SCREEN Events . . . . . .  0
    AOF TOD Events. . . . . .  0
    AOF Rule Execution Summary Statistics
    AOF COMMAND rules executed. . . . . 22
    AOF rule DISABLE (TERM sections) exec . . . . . .  3
    AOF DOM rules executed. . . . . .  0
    AOF rule ENABLE (INIT sections) exec. . . . . .  3
    AOF EOM rules executed. . . . . .  1
    AOF GLV rules executed. . . . . .  0
    AOF MESSAGE rules executed. . . . . 1504
    AOF OMEGAMON rules executed . . . . . .  0
```

```
    AOF REQUEST rules executed. . . . . .  1
    AOF SECURITY rules executed . . . . . .150
    AOF SCREEN rules executed . . . . . .  0
    AOF TOD rules executed. . . . . .  0
6>                  OSFSUMM: OSF Summary Statistics
    Count of Queued Server Transactions . . . . . 24
    Count of Queued Trans With No Queue Time. . . . . .  0
                        OSF Queue Depth Statistics
    OSF Queue Depth 0   Occurrences . . . . . .  7
    OSF Queue Depth 1   Occurrences . . . . . .  2
    OSF Queue Depth 2   Occurrences . . . . . .  1
    OSF Queue Depth 3   Occurrences . . . . . .  2
    OSF Queue Depth 4   Occurrences . . . . . .  1
    OSF Queue Depth 5   Occurrences . . . . . .  1
    OSF Queue Depth 6   Occurrences . . . . . .  1
    OSF Queue Depth 7   Occurrences . . . . . .  2
    OSF Queue Depth 8   Occurrences . . . . . .  1
    OSF Queue Depth 9   Occurrences . . . . . .  1
    OSF Queue Depth 10  Occurrences . . . . . .  1
    OSF Queue Depth 11  Occurrences . . . . . .  1
    OSF Queue Depth 12  Occurrences . . . . . .  2
    OSF Queue Depth 13  Occurrences . . . . . .  1
    OSF Queue Depth 14  Occurrences . . . . . .  0
    OSF Queue Depth 15  Occurrences . . . . . .  0
    OSF Queue Depth 16  Occurrences . . . . . .  0
    OSF Queue Depth 17  Occurrences . . . . . .  0
    OSF Queue Depth 18  Occurrences . . . . . .  0
    OSF Queue Depth 19  Occurrences . . . . . .  0
    OSF Queue Depth >19 Occurrences . . . . . .  0
                      OSF Initiation/Termination Statistics
    Total Server Initiations. . . . . .  3
    OSFMIN Server Initiations . . . . . .  3
    OSFQADD Server Initiations. . . . . .  0
    Server Failure Restarts . . . . . .  0
    Number of Terminations due to OSFRECYCLE. . . . . .  0
    High Water Mark Server Count. . . . . .  3
                      OSF Execute Queue Statistics
    Average OSF Execute Queue Time. . . . . 14.08 secs
    Lowest  OSF Execute Queue Time. . . . . 2 ms
    Highest OSF Execute Queue Time. . . . . 35.73 secs
                      OSF Transaction Statistics
    Average OSF Transaction Time. . . . . 7.48 secs
    Lowest  OSF Transaction Time. . . . . 54 ms
    Highest OSF Transaction Time. . . . . 50.34 secs
    Count of Completed OSF Transactions . . . . . 24
```

```
7>                COMMAND: Command Summary Statistics
    NOACTION commands counter . . . . . . . . . . . . . . . . . 20
    ACCEPT commands counter . . . . . . . . . . . . . . . . . .  2
    REJECT commands counter . . . . . . . . . . . . . . . . . .  0
    Number of OSF console commands. . . . . . . . . . . . . . .  1
    Number of ECF console commands. . . . . . . . . . . . . . .  2
    Number of commands reissued (length). . . . . . . . . . . .  1
                  Product Console Allocation Statistics
    Number of console allocation failures . . . . . . . . . . .  0
    Number of console allocation timeouts . . . . . . . . . . .  0
    Number of successful console allocations. . . . . . . . . .  7
    Number times console allocation waited. . . . . . . . . . .  0
    Accumulated console allocation wait time. . . . . . . . . 0 ms
8>                PERMVAR: Permanent GLV Summary Stats
    Number of Permanent Global Variables. . . . . . . . . . .  538
    Maximum Number of Blocks (GLOBALMAX). . . . . . . . . . . 5,000
    High-Used Blocks. . . . . . . . . . . . . . . . . . . . . 2,148
    In-Use Blocks . . . . . . . . . . . . . . . . . . . . . .  980
    Free Blocks On Free Chain . . . . . . . . . . . . . . . . 1,168
    Free Areas On Free Chain. . . . . . . . . . . . . . . . . . 18
    Pages In Global Workspace . . . . . . . . . . . . . . . .  313
    Global Variable Updates . . . . . . . . . . . . . . . . . 43
    SYSCHK1 Checkpoint Interval . . . . . . . . . . . . . . . . 15
    SYSCHK1 Checkpoints . . . . . . . . . . . . . . . . . . . . 43
    SYSCHK1 Checkpoint Retries. . . . . . . . . . . . . . . . .  0
    Global Variable Error Messages. . . . . . . . . . . . . . .  0
9>              TEMPVAR: Temporary GLV Summary Stats
    Number of Temporary Global Variables. . . . . . . . . . . 25
    Maximum Number of Blocks (GLOBALTEMPMAX). . . . . . . . . 5,000
    High-Used Blocks. . . . . . . . . . . . . . . . . . . . . 35
    In-Use Blocks . . . . . . . . . . . . . . . . . . . . . . 35
    Free Blocks On Free Chain . . . . . . . . . . . . . . . . .  0
    Free Areas On Free Chain. . . . . . . . . . . . . . . . . .  0
    Pages In Temporary Global Workspace . . . . . . . . . . .  313
    Temporary Global Variable Updates . . . . . . . . . . . . 68
    Temporary Global Variable Error Messages. . . . . . . . . .  0
10>              OPSVALUE: OPSVALUE Summary Statistics
    Number of OPSVALUE calls. . . . . . . . . . . . . . . . .  139
    Number of cross system OPSVALUE calls . . . . . . . . . . .  0
    Number of internal OPSVALUE calls . . . . . . . . . . . .  400
    Internal OPSVALUE unknown type calls. . . . . . . . . . . .  0
    Internal OPSVALUE SQL calls . . . . . . . . . . . . . . .  384
    Internal OPSVALUE STATEMAN calls. . . . . . . . . . . . . 11
    Internal OPSVALUE TOD CATCHUP calls . . . . . . . . . . .  1
    Internal OPSVALUE GLVEVENT cleanup calls. . . . . . . . .  0
    Internal OPSVALUE GLVJOBID cleanup calls. . . . . . . . .  4
```

```
11>                     SQL: SQL Summary Statistics
                        SQL Statements Executed
    Total number of SQL statements executed . . . . . .  116
    CREATE TABLE statements . . . . . .  2
    INSERT statements . . . . . . 12
    UPDATE statements . . . . . . 17
    SELECT statements . . . . . . 83
    DELETE statements . . . . . .  2
    DECLARE CURSOR statements . . . . . .  0
    OPEN statements . . . . . .  0
    FETCH statements. . . . . .  0
    CLOSE statements. . . . . .  0
    DROP TABLE statements . . . . . .  0
    ALTER TABLE statements. . . . . .  0
                    SQL Engine Internal Operations
    Direct reads. . . . . . 24
    Sweep reads . . . . . .  561
    Insertions. . . . . . 12
    Writes (Updates). . . . . . 49
    Deletions . . . . . .  1
    Compile errors. . . . . .  0
    Execution errors. . . . . .  2
    Logic errors. . . . . .  0
                    SQL Execution Timings
    Statements compiled . . . . . . 36
    Average compilation time. . . . . . 1 ms
    Statements executed . . . . . .  116
    Average execution time. . . . . . 5 ms
12>                       EPI: EPI Statistics
    Total count of Address EPI commands . . . . . .  366
    LIST commands . . . . . . 96
    DELETE commands . . . . . .  0
    DEFINE commands . . . . . .  2
    CHANGE commands . . . . . .  2
    LOGON commands. . . . . . 11
    LOGOFF commands . . . . . .  8
    INQuire INPUT commands. . . . . . 32
    TYPE commands . . . . . . 25
    RDSCRN commands . . . . . . 46
    RDSCRN/ROW commands . . . . . .  0
    RDCURSOR commands . . . . . . 25
    BIND commands . . . . . .  0
    UNBIND commands . . . . . .  0
    ENQ  commands . . . . . .  0
    DEQ commands. . . . . .  0
    CLEANUP requests. . . . . . 92
    Miscellaneous commands. . . . . . 27
```

The following list describes the major segments in the previous report:

1 GENERAL Segment

Contains general statistics related to the execution of CA OPS/MVS such as product start time and stop time.

Specify SEGMENTS=(GENERAL) and RECTYPE=(SUMMARY) to report only this segment.

2 MESSAGE Segment

Contains statistics related to WTO and WTOR message processing.

Specify SEGMENTS=(MESSAGE) and RECTYPE=(SUMMARY) to report only this segment.

3 PROCBLK Segment

Contains statistics related to CA OPS/MVS process block usage. Process blocks represent virtual storage used to process events intercepted by the product. The number of process blocks and the size of each block are controlled by various product parameters (for example, PROCESS, AOFSIZE).

Specify SEGMENTS=(PROCBLK) and RECTYPE=(SUMMARY) to report only this segment.

4 EVNTEXIT Segment

Indicates the number of call occurrences for various system and subsystem interface points.

Specify SEGMENTS=(EVNTEXIT) and RECTYPE=(SUMMARY) to report only this segment.

5 AOFSUMM Segment

Contains statistics related to AOF events. This segment indicates the number of times that specific event types occurred and were eligible for rules processing. This segment also indicates the number of times that rules executed for specific event types.

Note: Since an event can trigger multiple rules, the number of executions can be greater than, less than, or equal to the number of events.

Specify SEGMENTS=(AOFSUMM) and RECTYPE=(SUMMARY) to report only this segment.

6 OSFSUMM Segment

Contains statistics related to the OSF component. This segment reports various OSF server transaction counts, OSF queue depth frequencies, and OSF queueing and execution values.

Specify SEGMENTS=(OSFSUMM) and RECTYPE=(SUMMARY) to report only this segment.

7 COMMAND Segment

Contains statistics related to command processing. This segment reports command disposition statistics such as rejected or accepted. The Product Console Allocation Statistics section reports on the various extended, migration, and subsystem consoles used by the product to issue commands and capture responses.

Specify SEGMENTS=(COMMAND) and RECTYPE=(SUMMARY) to report only this segment.

8 PERMVAR Segment

Contains statistics related to permanent global variable processing. Permanent global variables are checkpointed in the SYSCHK1 VSAM linear data set, and they include global variables and relational tables.

Specify SEGMENTS=(PERMVAR) and RECTYPE=(SUMMARY) to report only this segment.

9 TEMPVAR Segment

Contains statistics related to temporary global variable processing. Temporary global variables are those variables that are not saved across product restarts and that include the job ID and event classes of variables and the rows of GLOBAL TEMPORARY relational tables.

Specify SEGMENTS=(TEMPVAR) and RECTYPE=(SUMMARY) to report only this segment.

10 OPSVALUE Segment

Contains statistics related to the OPSVALUE OPS/REXX function and the internal product facilities that use this interface for global variable and RDF (SQL) processing.

Specify SEGMENTS=(OPSVALUE) and RECTYPE=(SUMMARY) to report only this segment.

11 SQL Segment

Contains statistics related to the CA OPS/MVS SQL facility. This segment reports SQL statement counts, internal operations counts, error counts, and various execution timings.

Specify SEGMENTS=(SQL) and RECTYPE=(SUMMARY) to report only this segment.

12 EPI Segment

Contains statistics related to the External Product Interface (EPI) facility.

This segment reports counts of various ADDRESS EPI host commands and the EPI-related command processors.

Note: SESSCMD is internally converted to a number of ADDRESS EPI commands.

Specify SEGMENTS=(EPI) and RECTYPE=(SUMMARY) to report only this segment.

Generate the AOFEVENT Segment

The statistics reported in the AOFEVENT Segment of the Automation Statistics Report are accumulated throughout the life of a particular AOF rule. When the rule is disabled and the SMFRULEDISABLE parameter is set to YES, CA OPS/MVS generates an SMF record that describes the life of the AOF rule. The AOFEVENT segment reports the data in this SMF record.

To generate the AOFEVENT Segment, specify SEGMENTS=(AOFEVENT) and RECTYPE=(EVENT).

Example: AOFEVENT Segment

Here is an example of the AOFEVENT Segment for three different rules:

```
             Automation Measurement Environment
Statistics For SMFID S034    First Record Start: 2009/08/09 14:36:10
SSNM OPSS                       Last Record End:            14:36:11
                        AOFEVENT: AOF Event Statistics
                           General Rule Information
1>    Rule type . . . . . . . . . . . . . . . . . . . . .  Command
      Ruleset name. . . . . . . . . . . . . . . . . . . . .  0
      Rule name . . . . . . . . . . . . . . . . . . . . CMDCOUNT
2>    Rule enablement time and date . . . . .  2009/08/08 12:30:05
3>    Total time rule enabled . . . . 1 days 2 hours 6 mins 5 secs
      Message ID or other criterion . . . . . . . . . . . . . .  *
4>    Rule Section Execution Counts
      Initialization section executions . . . . . . . . . . . .  1
      Processing section executions . . . . . . . . . . . . .  224
      Termination section executions. . . . . . . . . . . . .  1
      Number of NOOPSLOG executions . . . . . . . . . . . . .  0
                           Rule Firing Statistics
      Rule firing limit . . . . . . . . . . . . . . . . . 10,000
      Rule firing high water mark . . . . . . . . . . . . . . 39
5>                 Rule Execution Failure Statistics
      Last failure type . . . . . . . . . . . . . . . . . . . .
      Number of execution failures. . . . . . . . . . . . . .  0
      Abend code or REXX execution error. . . . . . . . . . .  0
      Last execution failure time . . . . . . . . . . . . <none>
      Last execution failure message ID . . . . . . . . . . . .
```

```
6>                    General Rule Information
      Rule type . . . . . . . . . . . . . . . . . . . . Security
      Ruleset name. . . . . . . . . . . . . . . . . . .  SEC
      Rule name . . . . . . . . . . . . . . . . . . . . ARCHSECG
      Rule enablement time and date . . . . .  2009/08/08 12:30:05
      Total time rule enabled . . . . 1 days 2 hours 6 mins 5 secs
      Message ID or other criterion  OPSGLOBALGLOBAL0.ARCH_TRACK.*
                        Rule Section Execution Counts
      Initialization section executions . . . . . . . . . . . .  1
      Processing section executions . . . . . . . . . . . . . . 14
      Termination section executions. . . . . . . . . . . . . .  1
      Number of NOOPSLOG executions . . . . . . . . . . . . . .  0
                        Rule Firing Statistics
      Rule firing limit . . . . . . . . . . . . . . . . . . 10,000
      Rule firing high water mark . . . . . . . . . . . . . .  0
      Rule Execution Failure Statistics
      Last failure type . . . . . . . . . . . . . . . . . . . .
      Number of execution failures. . . . . . . . . . . . . .  0
      Abend code or REXX execution error. . . . . . . . . . .  0
      Last execution failure time . . . . . . . . . . . . . <none>
      Last execution failure message ID . . . . . . . . . . . .
7>                    General Rule Information
      Rule type . . . . . . . . . . . . . . . . . . . .  Message
      Ruleset name. . . . . . . . . . . . . . . . . . . . VTAM
      Rule name . . . . . . . . . . . . . . . . . . . .  IST530I
      Rule enablement time and date . . . . .  2009/08/08 12:30:02
      Total time rule enabled . . . . 1 days 2 hours 6 mins 8 secs
      Message ID or other criterion . . . . . . . . . .  IST530I
                     Rule Section Execution Counts
      Initialization section executions . . . . . . . . . . . .  1
      Processing section executions . . . . . . . . . . . . 16,372
      Termination section executions. . . . . . . . . . . . . .  1
      Number of NOOPSLOG executions . . . . . . . . . . . . 16,372
                        Rule Firing Statistics
      Rule firing limit . . . . . . . . . . . . . . . . . . 10,000
      Rule firing high water mark . . . . . . . . . . . . . . 44
                   Rule Execution Failure Statistics
      Last failure type . . . . . . . . . . . . . . . . . . . .
      Number of execution failures. . . . . . . . . . . . . .  0
      Abend code or REXX execution error. . . . . . . . . . .  0
      Last execution failure time . . . . . . . . . . . . . <none>
      Last execution failure message ID . . . . . . . . . . . .
```

The following list describes the important items in the above report:

1 Rule Type

Displays the type of rule. For example, command or security.

2 Rule Enablement

Contains the date and time the rule was enabled.

3 Total Time

Contains the total time that the rule was enabled.

4 Rule Section Execution Counts

Contains the execution counts of the various sections in the rule.

5 Rule Execution Failure Statistics

Contains the statistics related to execution failures that occurred for the rule.

6 General Rule Information

Contains statistics for the second rule.

7 General Rule Information

Contains statistics for the third rule.

Generate the OSFEVENT Segment

The statistics reported in the OSFEVENT Segment of the Automation Statistics Report describe a particular OSF server transaction. When the OSF server transaction finishes and the OSFTRANSSMFREC parameter is set to YES, CA OPS/MVS generates an SMF record that describes the transaction. The OSFEVENT Segment reports the data in this SMF record.

To generate the OSFEVENT Segment, specify SEGMENTS=(OSFEVENT) and RECTYPE=(EVENT).

Example: OSFEVENT Segment

Here is an example of the OSFEVENT Segment:

```
                Automation Measurement Environment
  Statistics For SMFID S034    First Record Start: 2005/11/11 07:45:19
  SSNM OPSS                       Last Record End:           14:28:17
                       OSFEVENT: OSF Transaction Statistics
                            OSF Transaction Statistics
1>      Server job name . . . . . . . . . . . . . . . . . . . OPSOSF
        Server step name. . . . . . . . . . . . . . . . . . OPSS010E
        Server procstep name. . . . . . . . . . . . . . . . . . OPSS
        Server ASID . . . . . . . . . . . . . . . . . . . . . . 010E
2>      Command text  . . . . . . . SUBMIT 'OPS.O.CCLXCNTL(ARCHJOB)'
        Command length. . . . . . . . . . . . . . . . . . . . . . 28
        Command count . . . . . . . . . . . . . . . . . . . . .  6
        Average command output lines. . . . . . . . . . . . . .  3
        Average command I/O count . . . . . . . . . . . . . . . 22
3>      Average command elapsed time. . . . . . . . . . . . . 2 secs
4>      Average command CPU (TCB + SRB) time. . . . . . . . . 193 ms
5>      Average time spent on server queue. . . . . . . . . .   2 ms
```

The following list describes the important items in the above report:

1

The job name of the OSF server that processed the transaction.

2

The text of the command that was sent to the OSF server that initiated the processing of the transaction.

3

The average elapsed time that the OSF server took to process the transaction.

4

The average CPU time that the OSF server took to process the transaction.

5

The average amount of time this transaction waited on the internal OSF Execute Queue until a server became available to process the transaction.

Generate the OSFTERM Segment

The statistics reported in the OSFTERM Segment of the Automation Statistics Report are accumulated throughout the life of a particular OSF server. When the server is terminated, CA OPS/MVS generates an SMF record that describes the life of the server. The OSFTERM Segment reports the data in this SMF record.

To generate the OSFTERM Segment, specify SEGMENTS=(OSFTERM) and RECTYPE=(EVENT).

Example: OSFTERM Segment

Here is an example of the OSFTERM Segment:

```
                Automation Measurement Environment
  Statistics For SMFID S034    First Record Start: 2005/08/07 08:04:03
  SSNM OPSS                       Last Record End:            08:04:31
                      OSFTERM: OSF Transaction Statistics
                           General Server Information
1>      Server job name . . . . . . . . . . . . . . . . . . . OPSOSF
        Server step name. . . . . . . . . . . . . . . . . . OPSS0044
        Server procstep name. . . . . . . . . . . . . . . . . OPSS
        Server ASID . . . . . . . . . . . . . . . . . . . . . 0044
        OPOSEX flags. . . . . . . . . . . . . . . . . . . . 11000000
2>                      TSO Transaction Statistics
        Wait Time Limit Per Transaction . . . . . . . . . . .  120
        Maximum Elapsed Time Per Transaction. . . . . . . . .  120
        Maximum CPU Seconds Per Transaction . . . . . . . . . . 15
        Maximum Output Lines Per Transaction. . . . . . . .  1,000
        Total OSF Lines Output. . . . . . . . . . . . . . . . . 39
3>      Total OSF Transaction Count . . . . . . . . . . . . . . 10
```

The following list describes the important items in the above report:

1

Contains the job name of the OSF server that processed the transaction.

2

Contains the statistics related to TSO transactions. The first part of this section contains the values of various OSF parameters that control server operation. These values represent the settings of these parameters at the time the server terminated.

3

Contains the number of transactions processed by the OSF server.

Generate the IMS Segment

The statistics reported in the IMS Segment of the Automation Statistics Report describe the execution of IMS Batch Message Processing (BMP) subtasks. Each IMS BMP is reported as a separate set of statistics.

To generate the IMS Segment, specify SEGMENTS=(IMS) and RECTYPE=(EVENT).

Example: IMS Segment

Here is an example of an IMS Segment:

```
                Automation Measurement Environment
   Statistics For SMFID S034    First Record Start: 2005/08/07 08:04:03
   SSNM OPSS                       Last Record End:            08:04:31
                            IMS: IMS Event Statistics
                                  BMP Statistics
       CA OPS/MVS subsystem ID. . . . . . . . . . . . . . . OPSS
1>     IMS ID. . . . . . . . . . . . . . . . . . . . . . . . IVP1
       Time BMP waiting for work . . . . . . . . . . 6 mins 18 secs
      Time BMP processing work. . . . . . . . . .  3.55 secs
2>     Average command processing time . . . . . . . . .  0.71 secs
       Commands processed. . . . . . . . . . . . . . . . . .  5
       Display-only commands processed . . . . . . . . . . .  5
3>     Failed commands . . . . . . . . . . . . . . . . . . .  0
       Output lines. . . . . . . . . . . . . . . . . . . . .  199
4>     CA OPS/MVS subsystem ID. . . . . . . . . . . . . . .  OPSS
       IMS ID. . . . . . . . . . . . . . . . . . . . . . . . IVP2
       Time BMP waiting for work . . . . . . . . . . 9 mins 10 secs
       Time BMP processing work. . . . . . . . . . . .  5.55 secs
       Average command processing time . . . . . . . . .  1.11 secs
       Commands processed. . . . . . . . . . . . . . . . . .  5
       Display-only commands processed . . . . . . . . . . .  4
       Failed commands . . . . . . . . . . . . . . . . . . .  0
       Output lines. . . . . . . . . . . . . . . . . . . . .  308
```

The following list describes the important items in the above report:

1

Contains the IMS ID that is controlling the IMS BMP subtask.

2

Contains the average command processing time for all commands processed by this BMP. It is computed by dividing the total elapsed time spent by this BMP processing commands by the number of processed commands.

3

Contains the number of commands that failed in the BMP.

4

Contains statistics for each IMS BMP.

Appendix A: Supplied Sample Rules and Programs

This section contains the following topics:

Available Sample AOF Rules and OPS/REXX Programs

CA OPS/MVS distributes sample AOF rules and OPS/REXX programs that demonstrate various aspects of automating operations in a mainframe environment. These procedures adhere to the specific environment where they have been created and tested, and depending on the desired outcome, the provided automation may need to be modified from site to site.

The primary purpose of these rules and programs is to do the following:

- Outline and demonstrate automated techniques that are needed across all data centers
- Illustrate the effective usage of the automation tools that CA OPS/MVS provides, such as the OPS/REXX host environments, OPS/REXX functions, global variables, local variables, and much more
- Provide the syntax of these applications that you can use as references and templates or starting points when you implement site-specific automation

How to Locate Supplied Sample Rules and OPS/REXX Programs

Many of the supplied sample rules and programs work together as an application to automate a specific system component or event.

You can locate these sample rules and programs in the following data sets, which CA OPS/MVS creates during installation:

- *h*lq.CCLXRULS

  Contains the sample AOF rules.

  Important! Do not auto enable or enable the entire sample rule data set. You must first carefully review each sample rule to determine its applicability to your system. If the sample rule is desirable within your environment, you should copy it to an existing production AOF ruleset.

- *h*lq.CCLXSAMP

  Contains the sample OPS/REXX programs.

All AOF rules and OPS/REXX program members for automated applications include the following:

- Comments that outline the logic flow of the application
- Descriptions of the various automation tools that the application uses
- Steps needed to implement the sample rule and/or OPS/REXX program

The following sections of this appendix will categorize the specific component or system event that the supplied sample rules and programs address. Each individual sample rule and OPS/REXX program utilized within these applications are listed in Appendixes B and C.

CA OPS/MVS Components

Several AOF rules and OPS/REXX programs demonstrate creating automation to utilize, control, and monitor various components within CA OPS/MVS.

These include:

- AOF
- API
- EPI
- HWS
- Opslog
- OSF
- SOF
- SSM

AOF Component

The following applications provide automation for the AOF Component of CA OPS/MVS.

TIMECHNG

Resynchronizes AOF TOD rules during dynamic Daylight savings time changes. No IPL is performed.

ZEROAOF

The AOF ZEROAOF request rule demonstrates a programmatic method of obtaining, displaying, and saving AOF rule information for any enabled rule that has a zero fire count. While this logic may provide some drop-in value, it primarily demonstrates the programmatic manipulation that can be performed against AOF statistical information across cycles of CA OPS/MVS.

API Component

The following applications provide automation for the API Component of CA OPS/MVS.

APIHRTB1

Issues a highlighted banner alert message in response to a warning or problem heartbeat issued by a participating CA product.

APIHRTB2

Processes a message issued from the CA OPS/MVS monitor task that indicates a participating CA product has not generated any heartbeat events within an interval determined by that CA product. This is useful if the CA product is in a state that is causing it to not generate any normal heartbeat events.

APIHRTB3

DOMs previously issued heart beat interval failures upon participating CA products re-issuing NORMAL API heart beat events.

SSMCAAPI

Provides general active status processing for any participating CA product. This API communicates a product's active status (STARTING, UP, STOPPING, or DOWN) to CA OPS/MVS, facilitating a common method for SSM to capture the current active state of any CA mainframe product. The rule named SSMCAAPI is provided to capture these events and communicate to SSM. You must enable this rule for it to be used by SSM.

HWS Component

The following application provides automation for the HWS Component of CA OPS/MVS.

APIHWSV

Formats and issues HWS hardware event data (variables) as a multi-line WTO (MLWTO).

OPSLOG Component

The following applications provide automation for the OPSLOG Component of CA OPS/MVS.

OPS34450

Automatically switches to another eligible active OPSLOG when it detects that the current live OPSLOG is approaching the maximum internal message number limit. This eliminates the need to restart CA OPS/MVS after deleting and reallocating the OPSLOG.

OPSLGEXT

Demonstrates utilizing the OPS/REXX OPSLOG() function. This function can extract data from the CA OPS/MVS OPSLOG using filter criterias, such as by jobname, asid, or message ID. This type of automation may be useful in collecting diagnostic data for some problem jobname that is occurring on a system, and then forwarding this data to the responsible support teams to assist them with the debugging effort.

OPSLGSCN

Periodically performs the following functions:

- Scans the OPSLOG looking for specific OPS/REXX compiler and execution error messages that might have occurred within rules and/or programs.
- Generates an alert email of these messages to a list of designated users.

SECWEBV1, SECWEBV2, and SECWEBV3

Demonstrates setting up security within the OPSLOG Webview component.

OSF Component

RESETOSF

End-User OPS/REXX pgm to stop or force CA OPS/MVS servers and reset (delete) all server requests in the specified server queue.

SOF Component

The following applications provide automation for the SOF Component of CA OPS/MVS.

SOFCMDR, SOFSECR, SOFCMDU, SOFSECU

Secures console commands that control the SOF component.

OPSOSF

Creates a pseudo command rule to simulate SOF commands from a console command.

OPSOSF001

Varies devices offline or online as detected by the SOF API event.

SSM Component

The following applications provide automation for the SSM Component of CA OPS/MVS.

OP4UEXIT

User command exit allows user to trigger any type of end-user automation against some SSM resource. Users can implement their own new line and primary command to be able to invoke a program. For example, an OPS/REXX validation program, SSM Note or a new ISPF application.

SECSSM1, SECSSM2, and SECSSM3

Demonstrate how to secure the updating of SSM components.

SSMALTSB

Demonstrates how to alter normal SSM subrequisite processing to allow for a different controlled shutdown of a group of resources using the XSUBREQ action process.

For example, startup may be JOBA, JOBB, JOBC, and then JOBD. Upon shutdown the normal SSM subrequisite processing would be to stop JOBD, JOBC, JOBB, and then JOBA. This sample demonstrates how to alter the normal subrequisite shutdown so that JOBC stops first, JOBA stops second, JOBD stops third, and JOBB stops last.

SSMCHECK

Implements a health check against SSM after IPL to ensure that any failed or problem resource has been acted upon.

SSMCNTL

Demonstrates creating a console interface application to control and monitor SSM functionality.

SSMEOJ

Demonstrate implementing end-user logic into SSM to monitor and control batch jobs similar to SSMEOM that monitors STCs.

SSMEXCPS

Displays all SSM STCTBL resources for all CA OPS/MVS MSF connections where both CURRENT and DESIRED states <> UP, or the CURRENT state is down.

SSMMAINT

Demonstrates how to dynamically implement SSM table changes during an IPL or a recycle of CA OPS/MVS.

SSMMAINT uses the following process to implement the SSM table changes:

1. Creates backups of existing production tables
2. Loads and uses the predefined new tables

SSMMOVE, SSMPLEXC, SSMXPREQ, and SSMXSUBR

Illustrates one method of monitoring and configuring cross-system dependencies within a sysplex.

SSMQUERY

Demonstrates the logic needed to identify the table where the SSM resource resides, for SSM configurations that have STCs in multiple resource tables.

SSMTREE

Generates a formatted file representation of a system state managed resource table. This file can then be used with GRAPHVIZ http://www.graphviz.org/License.php to display a graphical representation of the table's resource relationships.

SSMWEBSP

Provides rules and procedures to control a deployment manager WebSphere configuration within SSM.

SSMXCHCK

Determines the state of a remotely monitored SSM resource within the MSF connected or sysplex environment.

SSM2XCEL

Insert SSM resource and action table data into a sequential dsn so that it can be FTP'd and viewed as an Xcel document.

CA Products

This section provides specific system components or events automated by the supplied samples for CA products.

CA Datacom

The following applications provide CA Datacom environment automation.

SYSVDTCM

Utilizes the CA SYSVIEW interface to obtain the active CA Datacom ASIDs to determine if any of the data areas for each CA Datacom exceeds the defined percentage value. If a threshold is exceeded, an alert is generated.

SHUTDTCM and STRTDTCM

Initiates the start up and shutdown of a master CA Datacom/AD Multi User Facility region that is being used within a CA 11 environment.

CA IDMS

IDMSAREA

Invoke and manipulate the collected DCMT D AREA command output. If the LOCK status is in ‘OFL’ for any area then send out an alert message.

CA MIM

The following applications provide CA MIM environment automation.

APIMIMGR

Responds to problem related CA MIM API events.

The MIM2211 API event: This API event triggers when CA MIM detects a delay to a VARY ONLINE/OFFLINE request to a tape drive. The default logic within this API rule allows the VARY ONLINE/OFFLINE request to pend for 90 seconds before aborting back to CA MIM. This allows CA MIM across the MIMplex to continue processing.

The MIM2225 API event: This particular API event is triggered when Vary commands initiated by CA MIM are requeued because IEEVARYD is unable to obtain a SYSIEFSD ENQ within 5 seconds.

MIAIBR14

Causes tape volume dismounts to occur under CA MIA GLOBAL Tape Device Allocation Serialization for Tape Volumes that remain mounted in unallocated MIA-managed tape devices.

MIMQUERY

Creates a utility that TSO users can invoke to obtain and view the status of a specific QNAME or RNAME enqueue resource within a CA MIM MIMplex environment.

MIMTAPE

Reports on outstanding tape mounts for CA MIM managed tape devices. It replaces the existing TAPEMNT* sample rules.

MEDSMIM

The Mainframe Environment Discovery Service MIM (MEDSMIM) is an environmental reporting application used by CA MIM customers. This diagnostic tool greatly reduces the time needed to gather CA MIM complex environmental information. It can also regularly ensure your CA MIM address spaces are setup and running optimally.

CA PDSMAN

The following provides CA PDSMAN environment automation.

APIPDSMN

Responds to various CA PDSMAN API events, including LLA out of synch conditions, space, thresholds, and invalid libraries.

CA Process Automation

The following provides CA Process Automation environment automation.

PAZ*

Demonstrate the extension of CA OPS/MVS SSM functionality to the distributed environment by controlling SAP, running on Linux, through CA Process automation. The functions of these sample applications are invoked using SSM actions as specified in the action table.

CA Scheduler

The following provides CA Scheduler environment automation.

APISCHED

Provides sample API rules that generate alerts when it processes the CA Scheduler events SCHDJABEND, SCHDJLATE, and SCHDJFAIL.

For SCHDJABEND events, additional logic demonstrates how to restart or cancel a specific job based on the number and type of abends.

CA SYSVIEW

The following applications provide CA SYSVIEW environment automation.

APISYSVC and SYSVCICS

Outlines and demonstrates the process of creating an effective CA OPS/MVS application that monitors and responds to CICS threshold alerts collected by CA SYSVIEW. This application provides a foundation of implementing a more granular automated decision making application that is needed when processing these CA SYSVIEW alerts.

DB2WLMCK

Utilizes the CA SYSVIEW interface to identify and activate failed DB2 WLM applications.

GSVXSSM*, GSVXLINE, GSVXUSER

Demonstrates the basic code needed to create a CA SYSVIEW REXX program that manipulates and displays CA OPS/MVS SSM data. The primary purpose of this sample REXX program is to demonstrate utilizing the CA SYSVIEW RXDISP command to manipulate CA OPS/MVS data. This sample specifically manipulates the SSM component of CA OPS/MVS. The out-of-the-box functionality will allow for specific SSM resource monitoring and control from within CA SYSVIEW.

SPOOLMON

Utilizes the CA SYSVIEW interface to identify the top spool user when the JES2 TGS spool warning level has been exceeded.

SYSVALRT

You can create proactive automation between the CA OPS/MVS and CA SYSVIEW interface that alerts on and resolves potential problem ASIDs before they impact system performance.

This sample application monitors and responds to system alerts collected by CA SYSVIEW. This application provides automated decision making when processing these alerts.

SYSVCHCK

Demonstrates a coding technique you can use to obtain and interrogate various CA SYSVIEW related data and generate SMTP email alerts for any exceeded defined thresholds.

The specific CA SYSVIEW data you can obtain include:

- Coupling facility related data
- MQ related data
- Page data set related data
- WLM related data

SYSVCTDQ

Invoke the CA SYSVIEW CICS CTDATA GLOBAL command to obtain transient data queues for all active CICS regions. This application also generates an alert for any TDQ that that has a current QCount greater than a defined threshold.

SYSVDSNX

The logic within this OPS/REXX program demonstrates the basic code needed to extract data set extent usage information for data sets active to a specific job. The CA SYSVIEW interface is used to obtain this information.

SYSVDTCM

Utilizes the CA SYSVIEW interface to obtain the active CA Datacom ASIDs to determine if any of the data areas for each CA Datacom exceeds the defined percentage value. If a threshold is exceeded, an alert is generated.

SYSVE

Illustrates creating a pseudo CMD rule to obtain CA SYSVIEW data from a console command.

SYSVIEWE

Demonstrates the basic OPS/REXX code needed to extract CA SYSVIEW data using the ADDRESS SYSVIEWE OPS/REXX host environment. This program uses the XVEXTRAC CA SYSVIEW command.

SYSVINIT

Determine if a batch jobs execution or clocktime is exceeding defined run times based on the initiator class in which it is running.

SYSVMSU

Process MSU4HAVG threshold alerts (Averages MSU usage). For warning/problem alerts, output from the ACTSUM AVERAGE CA Sysview command will be obtained and emailed through SMTP to a specific list of email IDs. An 'All OK' email will be generated when the MSU4HAVG alert returns a 'NORMAL' status.

SYSVOUTP

Uses the CA SYSVIEW LISTFILE command to extract spooled output of a job that is not being sent to the operator and therefore is not seen by normal means for CA OPS/MVS to process.

CA Workload Automation (CA WAEE)

WAEE2OPS

Demonstrate a technique that can be used to forward CA Workload Automation EE (CA WAEE) job monitoring data to CA OPS/MVS for further automated processing. Refer to sample member WAEE2OPS for complete implementation details.

CA XCOM

The following applications provide CA XCOM environment automation.

XCOPSEOJ and XCOPMSG

Initiates CA XCOM failover procedures in the event CA XCOM terminates abnormally.

CA DB2 DBM Products

The following application provides automation for CA DB2 DBM products.

PDT0170

Demonstrates logic needed to process data collector messages from the Execution Manager (Xmanager) component and invoke TSFBATCH Proc Unload to submit detector data to TSF.

CA Spool

The following provides CA Spool environment automation.

SHUTSPL

Initiates shutdown procedures for CA Spool print management.

CA TLMS

The following provides CA TLMS environment automation.

TLMSCHCK

Verifies that all components of the CA Dynam/TLMS product have successfully initialized.

CA 7

The following applications provide CA 7 environment automation.

CA7ABEND, CA7BRWSE, and CA7LATE

Demonstrates logic needed to process CA 7 browse log messages as collected using the CA 7 browse log interface. Specific logic browses CA 7 abend and late events.

SHUTCA7 and SHUTCA7I

Demonstrates logic needed to process CA 7 browse log messages as collected through the CA 7 browse log interface. Specific logic browses CA 7 abend and late events.

CA 11

The following provides CA 11 environment automation.

CA11MSG

Illustrates rule logic needed for message IDs that are longer than 10 characters, such as CA 11 generated messages. Sample logic processes and replies to the CA 11 shutdown WTOR.

Other Vendor Products

This section provides specific system components or events automated by the supplied samples for other vendor products.

CICS

The following applications provide CICS automation.

CICSEOJ

Monitors CICS regions, using an end-of-job event, that are abnormally terminating within a non CA OPS/MVS SSM environment. Demonstrates how to use EOJ rules.

CICSMON1, CICSMON2, CICSMON3

This application does the following:

- Monitors CICS regions that are abnormally terminating within a non CA OPS/MVS SSM environment.
- Demonstrates logic needed to correlate different events from the same ASID (msg1,msg2,eom, and so on) using unique global variables.

CICSREGS

Saves all CICS regions in an OPS/MVS variable. This data will be used during the aggressive SSM system shutdown sample procedures to invoke CICS shutdown procedures for non SSM CICS regions.

DFHS0126

Creates notification alerts for CICS regions that produce recursive MAXSOCKET reached conditions within a defined threshold period. (x times in y minutes.)

SHUTCICS

Demonstrates logic that can be used to initiate a controlled shutdown of a CICS region.

DB2

The following applications provide DB2 automation.

DB2MSTRS

Saves all DB2 master ASIDs in an CA OPS/MVS variable. This data will be used during the aggressive SSM system shutdown sample procedures to invoke DB2 shutdown procedures for non SSM DB2 regions.

DB2LOGDB

Collect the output of a DB2 command, specifically the DIS DB(dbname) command and store output in a sequential data set.

DB2TASKS

Saves all DB2 children tasks, which are internally started through the master DB2, into a CA OPS/MVS global variable.

The aggressive SSM system shutdown sample procedures use this data to allow the logic not to process non SSM controlled DB2 children tasks. Terminating the master DB2 region stops these tasks internally.

DB2THRD

Invoke the DIS THREAD(*) command and manipulate the output, canceling any thread with a non-zero token count.

DB2WLMCK

Utilizes the CA SYSVIEW interface to identify and activate failed DB2 WLM applications.

DSNT378I

Responds to DB2 IRLM lockouts caused by a system or DB2 failure within a shared data group.

DSNT376I and DSNT501I

Process DB2 Thread time out events and generate a single alert email with data from both related events.

DSNL027I

Suppress and deletes from syslog specific DB2 Distributed Agent abend failure MLWTO messages as determined by the last line of the MLWTO that indicates the failure reason.

DSNP002I

Processes DB2 data set definition failure MLWTO messages.

Alerts upon the first occurrence of each CSECT failure from the issuing DB2 master region. All other identical alerts of this failure for the same CSECT will wait 10 minutes before processing. This eliminates reacting to the same event more than once.

SHUTDB2

Performs shutdown of a DB2 region.

IMS

The following applications provide IMS automation.

IMSREGS

Saves all IMS regions in an OPS/MVS variable. This data will be used during the aggressive SSM system shutdown sample procedures to invoke IMS shutdown procedures for non SSM IMS regions.

SHUTIMS

Performs shutdown of an IMS region.

WebSphere MQ

The following applications provide WebSphere MQ automation.

SHUTMQS

Initiates a controlled shutdown of a WebSphere MQ region.

MQQCHK

Illustrates how to put and get messages on WebSphere MQ queues using the Address MQ host environment.

JES

The following provides JES automation.

JOBINFO

Obtains and displays specific detailed JES SSI 80 function call job data using the OPS/REXX OPSJESX().

JES2

The following provides JES2 automation.

JES2$TJ

Lets a user only perform a $TJ to a jobname that corresponds to the TSO user ID.

For example, user ID TSOUSR1 can only issue $TJ commands to TSOUSR1x jobnames.

SPOOLMON

Utilizes the CA SYSVIEW interface to identify the top spool user when the JES2 TGS spool warning level has been exceeded.

TSO

The following applications provide TSO automation.

TSOCHECK

Monitors TSO/E connectivity and response time.

IKT010D

Shuts down TSO even though TSO users are still logged on.

VTAM (Other Vendor Products)

The following application provides VTAM automation.

DVTMNODE

Demonstrates the logic needed to issue commands to the system and then collect and interrogate the command response, specifically the status of a VTAM node.

z/OS Activities

This section provides specific system components or events automated by the supplied samples for z/OS activities.

Disaster Recovery

Several AOF rules and OPS/REXX programs are used to demonstrate the automation of disaster recovery related activities.

The automated applications demonstrate initiating true disaster recovery scenarios and tests.

The applications:

- Address the issuing and interrogating of the commands needed to configure the storage subsystem.
- Outline the steps needed to create an SSM application to manage the started tasks that are started and monitored within a mirrored production and disaster recovery storage environment.

Note: For more information, see the DISASTER AOF rule.

Information Utilities

The following applications automate your information utilities.

CMDALL

Create a focal point of view for the command output of z/OS commands that is issued across many CA OPS/MVS MSF connected systems within the OPSLOG.

EMAILMSG

Create a focal application that can be called by any AOF rule or OPS/REXX program when a multi-line informational automated SMTP email needs to be generated.

EMAILTXT

Sends a single-line message as an SMTP email.

IPLINFO

Obtains various IPL data information, stores that data in a variable, and then issues a MLWTO OPSLOG only message with the data.

The MLWTO can be used as a quick tool within OPSLOG to locate the started IPLs, and the variable will be accessed using the OPSINQRY sample OPS/REXX program as a tool to present the collected IPL data across MSF connected systems.

OPSINQRY

Creates an end-user utility to issue commands or a series of commands and present the output in a scrollable response area.

SRCHLNKL

Determines if a module is found within any current link listed library.

SYSINFO

Obtains various system related IPL information for a system and saves it within the CA OPS/MVS shared variable data base. It then obtains this data from one focal point within TSO.

Checking ASID Existence on Remote Systems

The following applications provide automation for checking whether ASIDs exist on remote systems.

XJOBSTAT

Processes a pseudo sysplex routed command using the OPSTATUS() that can be used to remotely determine if an ASID is active.

XSYSASID

Determines if an ASID is active on a remote CA OPS/MVS MSF connected system, or within a specific sysplex system.

This OPS/REXX sample program can be called as an external function from another CA OPS/MVS OPS/REXX program when the status of an ASID on a remote system is needed. The sample logic assumes that the remote system is MSF connected, and a system within the local sysplex configuration.

Message Suppression and Manipulation

The following applications provide automation for the message suppression and message manipulation.

ACTNMSGS

Automating WTORs and action messages (descriptor codes 1, 2, 11) is an ongoing effort towards the goal of implementing aggressive system automation.

ACTNMSGS demonstrates a technique to:

1. Quickly identify these types of action messages in which a corresponding CA OPS/MVS rule is not logically processing them.
2. Use the OPSLOG to identify these events that are not currently being automated upon.
3. Create a rule to automate them.

IPLSUPPR

Improves message suppression at IPL time only if you are not utilizing aggressive suppression (as demonstrated within the SUPPRALL sample).

MLWTO*

Demonstrate various coding techniques needed to process MLWTO messages.

SUPPRALL

The AOF SUPPRALL rule details how you create a single suppression rule to suppress all messages except highlighted messages (WTORs, descriptor codes of 1, 2) and command responses.

WTOBUF

Clears console buffer backlogs.

Monitoring Batch Job Execution Times

SYSVINIT

Determine if a batch jobs execution or clocktime is exceeding defined run times based on the initiator class in which it is running.

Processing Cross-system Events

The following provides automation for processing cross-system events.

EOJXSYS

Demonstrates the logic that can be used to obtain status or event data from another system such as the maximum condition code (maxcc) of a batch job in order to perform some automated action on a local system.

Processing Job Enqueues

The following provides automation for processing job enqueues.

ENQCHECK

Monitors block times of system or job critical enqueues as reported through the D GRS,ANALYSE,BLOCKER command.

Processing Hardware Failures

The following provides automation for processing hardware failures.

IEC606I

Processes the first occurrence of a VTOC error condition from the specific DASD device within a sysplex. All other identical issuances of this failure for the same VTOC error will not be processed for the next 10 minutes. This prevents reacting to the same event more than once. Create dynamic JCL to invoke the ICKDSF utility to reinitialize the VTOC when the IEC606I event occurs.

Processing Problem ASIDs

The following applications automate the processing of your problem ASIDs.

ABENDLOG

Collects all IEF450I messages and stores in a RDF table. Create a dynamic TOD rule to trigger an OPS/REXX program to process this abend data and offload to a sequential data set. This application demonstrates an effective technique needed to collect and process frequently occurring events that require asynchronous actions such as storing to an external file.

IEA611I and IEA794I

Process IEA611I dump messages and generate an alert with the captured dump dsn for abends that are monitored from the IEA794I sample rule.

OPS44020

An address space identifier (ASID) may be looping and possibly generating excessive message traffic. The MSGDRAINRATE and MSGTHRESHOLD parameters detect address spaces that issue excessive message traffic. When this situation occurs, a message is issued that triggers rule member OPS44020 to generate an alert if a defined threshold is met.

Processing WTORs

The following applications provide WTOR automation.

WTORS

This sample pseudo command rule creates a quick and easy tool for a manual request of WTORS within a sysplex.

The sample utilizes the OPSTATUS() OPS/REXX function to return all WTOR information and reformat the display to include:

- Reply ID
- Text
- System
- Jobname
- Outstanding wait time for the WTOR

The jobname and wait time are unobtainable using the z/OS D R,L command.

MLWTOR

Sample code to demonstrate how to simulate a multi-line highlighted WTOR.

Processing z/OS Commands

The following applications provide automation for processing z/OS commands.

CMDAUTHW

Requests WTOR confirmation for a specific command or list of commands.

CMDVARY

Using CMDAUTHW as a template, CMDVARY requests WTOR confirmation for VARY commands when the range count for the specified range of devices is greater than the VARY command value. This safeguards against lengthy VARY commands from impacting the system.

Tape Mount Pendings

The following provides automation for pending tape mounts.

TAPEMNT1, TAPEMNT2, TAPEMNT3

Generates alert for outstanding tape mounts that exceed a defined threshold value.

For more information, see CA MIM sample MIMTAPE if you are using CA MIM to montor tape mounts.

USS Processes Management

The following provides automation for USS process management.

SSMUSS1

Demonstrates a coding technique that can be used to monitor and control USS daemon server processes such as INETD within System State Manager (SSM).

TCPIPUSS

Demonstrates a technique used to query the statuses of z/OS Unix processes associated to a particular ASID.

zFS File System

The following provides automation for the z/FS file system.

BPXI078D

Replies to WTORs generated by the OMVS zFS files system during initialization.

SHUTZFS

Initiates the shutdown of the OMVS zFS file system.

STARTZFS

Restarts the OMVS zFS file system after being stopped using the following command:

```
F OMVS,STOPPFS=ZFS command.
```

z/OS System IPL

The following application automates z/OS system IPL.

IPLTIME

Issues various system commands after a system IPL.

z/OS System Shutdown

The following application automates z/OS system shutdown.

SHUTSYS

These sample procedures demonstrate logic in implementing an aggressive system shutdown for both non SSM managed resources and SSM controlled resources.

Appendix B: Sample AOF Rules

This section contains the following topics:

Available Sample Rules (see page 669)

Available Sample Rules

This section lists the details regarding the individual sample AOF rules as outlined in the appendix Supplied Sample Rules and Programs, which overviews a specific automatable system component or event.

You can find these rules in *hlq*.CCLXRULS.

ABENDLOG

Rule Type: Message

Subsystem: z/OS

Collects and stores all IEF450I messages in a RDF table. Creates a dynamic TOD rule to trigger an OPS/REXX program to process the data and offload to a sequential data set.

APISVMVS

Rule Type: API

Subsystem: z/OS

Process CA Sysview MVS threshold API events

ACF8A900

Rule Type: Message

Subsystem: Security

Restarts the CA OPS/MVS security interface after CA ACF2 is active.

ACTNMSGS

Rule Type: Message

Subsystem: z/OS

Creates an automated filtering tool to assist in identifying WTORs and action messages (messages with descriptor codes 1,2,11) that have not been automated upon using a corresponding unique CA OPS/MVS rule.

AMALRCV

Rule Type: Message

Subsystem: z/OS

Sets the name of the CA NetMaster EPS Receiver ID parameter value.

APIHRTB1

Rule Type: API

Subsystem: n/a

Detects heartbeat failures for CA products that internally generate API CAHEARTBT events.

APIHRTB2

Rule Type: API

Subsystem: n/a

Detects abnormal heart beat intervals for CA products that internally generate API CAHEARTBT events.

APIHRTB3

Rule Type: API

Subsystem: n/a

Detects DOMs previously issued heart beat interval failures upon participating CA products reissuing NORMAL heart beat events.

APIMIMGR

Rule Type: API

Subsystem: n/a

Detects unique API events from CA MIM.

APIPDSMN

Rule Type: API

Subsystem: n/a

Demonstrates using an API rule for API events generated from CA PDSMAN.

APISCHED

Rule Type: API

Subsystem: n/a

Detects unique API events from CA Scheduler.

APISYSVC

Rule Type: API

Subsystem: z/OS

Processes CA SYSVIEW CICS lifetime transaction threshold API events.

APNOTIFY

Rule Type: Message

Subsystem: TSO

Sends the IKJ574I Broadcast Data Set Full message to a list of TSO user IDs. If the user does not respond after twelve messages, the CA Automation Point Notification Manager is used to escalate the problem.

ARMSAMP

Rule Type: Automatic Restart Manager

Subsystem: z/OS

Provides synchronization for System State Manager (SSM) any time ARM restarts an element that may be a resource managed by SSM.

ARMTASK

Rule Type: Message

Subsystem: z/OS

When z/OS issues message IEF403I for a started task initialization, this rule registers the task with z/OS Automatic Restart Manager, using the OPSARM function. The task name must be contained in GLOBAL0.ARMTASK.

ARMTASKE

Rule Type: Message

Subsystem: z/OS

Reacts when z/OS issues message IEF404I. If the task that is ending is defined to ARM, then OPSARM is issued to DEREGISTER the task.

ASOABEND

Rule Type: Message

Subsystem: z/OS

Traps z/OS symptom dump multiline message IEA995I and saves dump data in a relational table that you can review online using the table editor.

BMWRULE

Rule Type: Message

Subsystem: IMS

After starting IMS, you receive the outstanding REPLY:

```
*nn BMW001A REPLY DBDNAME TO STOP DATABASE.
```

The BMWRULE stops the database by REPLYing the DBDNAME to the BMW001A WTOR if *one* of the following occurs:

- Five occurrences of any mixture of the following two messages appear in any one minute between 08:00 and 17:00:
  - BMW002I dbdname LOGICAL ERROR
  - BMW003I I/O-ERROR ON DATABASE dbdname

BPXI078D

Rule Type: MSG

Subsystem: z/OS

Respond to initialization WTOR from OMVS zFS file system.

CA11MSG

Rule Type: API

Subsystem: n/a

Demonstrates creating MSG rules for message IDs greater than 10 characters (sample logic replies to CA 11 WTOR).

CANTSO

Rule Type: Request

Subsystem: TSO

Enables you to cancel TSO users and jobs that have the same user ID.

CAS9200I

Rule Type: Message

Subsystem: CAIENF

Restarts CA OPS/MVS when CAIENF is active.

CAS9300E

Rule Type: Message

Subsystem: CAIENF

Stops CA OPS/MVS when CAIENF is inactive.

CA7ABEND

Rule Type: Message

Subsystem: CA7

Processes CA 7 SMF0-19 job failure messages.

CA7BRWSE

Rule Type: Message

Subsystem: CA7

Processes additional segment lines for multi-segmented CA 7 events.

CA7LATE

Rule Type: Message

Subsystem: CA7

Processes CA 7 SCNP-11 job late messages.

CHE

Rule Type: Command

Subsystem: IMS

Shuts down the IOF BMP at the termination checkpoint.

CHKSYS

Rule Type: Message

Subsystem: CA OPS/MVS

Shows you how to create a generic INIT section in situations where a rule set is active on all systems, but certain rules are only intended to be ENABLEd on some systems. The generic INIT section can be used in any rule to maintain such rules.

CICSEOJ

Rule Type: EOJ

Subsystem: z/OS

Monitors CICS regions that are abnormally terminating within a non CA OPS/MVS SSM environment through an End-of-Job event. Demonstrates how to use EOJ rules.

CICSMON1,CICSMON2,CICSMON3

Rule Type: MSG,EOM

Subsystem: z/OS

Monitors CICS regions that are abnormally terminating within a non CA OPS/MVS SSM environment. Demonstrates logic needed to correlate different events from the same ASID (msg1,msg2,eom,etc) using unique global variables.

CICSREGS

Rule Type: Message

Subsystem: CICS

Saves the name of a CICS region within a CA OPS/MVS global variable. The variable is needed within the aggressive system shutdown sample procedure SHUTSYS2, to initiate the proper shutdown of a CICS region that may have started outside of STATEMAN.

CMDALLC

Rule Type: Command

Subsystem: z/OS

The CMD rule component of the CMDALL application. This command initiates the CMDALL OPS/REXX program.

CMDALLR

Rule Type: Request

Subsystem: z/OS

The REQ rule component of the CMDALL application. Triggered through the CMDALL OPS/REXX program on each system to issue desired command, collect response output, then send back to the requesting system.

CMDAUTHW

Rule Type: Command

Subsystem: z/OS

Requests WTOR confirmation for a specific command or list of commands.

CMDCOUNT

Rule Type: Command

Subsystem: z/OS

Keeps track of the number of times each z/OS and subsystem command has been used since the last IPL occurred. You can use this count to determine the types of commands being used most on the system, which will give you an idea of where automation of commands would be most valuable.

CMDVARY

Rule Type: CMD

Subsystem: z/OS

Forces WTOR verification if a VARY command exceeds a defined maximum range threshold.

DB2CMD

Rule Type: Command

Subsystem: DB2

Enables you to enter DB2 commands without suspending JES3. The command goes to an OSF server address space for execution.

DB2MSTRS

Rule Type: Message

Subsystem: DB2

Saves the name of a DB2 master region (ssidMSTR) within a CA OPS/MVS global variable. This variable is needed by the DB2TASKS sample rule which saves the starting of spawned DB2 regions. The variable is also needed within the aggressive system shutdown sample procedure, SHUTSYS2, to initiate the proper shutdown of a DB2 region that may have started outside of STATEMAN.

DB2TASKS

Rule Type: Message

Subsystem: DB2

Saves the names of DB2 tasks (ssidIRLM, ssidDIST, and so on.) that are spawned or started from a master DB2 subsystem within an OPS/MVS global variable. This variable is interrogated within the aggressive system shutdown procedures performed by the SHUTSYS2 OPS/REXX sample program. This variable allows the shutdown logic to bypass the stopping of a spawned DB2 tasks. These tasks are stopped through the shutdown of the associating DB2 master region.

DEBUGDOM

Rule Type: Delete-operator message

Subsystem: z/OS

Traps the DOM related z/OS and subsystem interface control blocks for debugging purposes. The control block information is saved in the CA OPS/MVS temporary global variables and can be viewed using OPSVIEW option 4.8.

DEBUGCMD

Rule Type: Command

Subsystem: z/OS

Traps the command related z/OS and subsystem interface control blocks for debugging purposes. The control block information is saved in the CA OPS/MVS temporary global variables and can be viewed using OPSVIEW option 4.8.

DEBUGWTO

Rule Type: Message

Subsystem: z/OS

Traps the z/OS and subsystem interface control blocks for debugging purposes.

DFHS0126

Rule Type: Message

Subsystem: N/A

Creates threshold (*x* times *y* seconds) monitoring logic for CICS max socket error conditions.

DFS690A

Rule Type: Message

Subsystem: IMS

Replies to the DFS690A message to allow the IOF BMP to wait for the control region to become active.

DFS8000A

Rule Type: Message

Subsystem: IMS

Highlights IMS MTO message ERROR CHECK TO MTO on the z/OS console any time the message is issued.

DFS994I

Rule Type: Message

Subsystem: IMS

Starts the IOF BMP only after the control region is fully active.

DISASTER

Rule Type: Command

Subsystem: z/OS

Initiates disaster recovery procedures.

DOM

Rule Type: Delete-operator message

Subsystem: z/OS

DOM events are triggered to remove high intensity messages from MCS consoles. This rule displays the WTO sequence number of the message being deleted.

DRL

Rule Type: Command

Subsystem: z/OS

Enables an operator to get a complete list of the outstanding replies using the old form of the D R,L command. The command will be converted to D R,L,CN=(ALL) if your release of z/OS requires this format.

DSNL027I

Rule Type: MSG

Subsystem: z/OS

Suppress and deletes from syslog specific DB2 Distributed Agent abend failure MLWTO messages as determined by the last line of the MLWTO that indicates the failure reason.

DSNP002I

Rule Type: MSG

Subsystem: z/OS

Process DB2 data set-definition failure MLWTO messages.

DSNT376I

Rule Type: Message

Subsystem: DB2

Process DB2 Thread timeout conditions.

DSNT378I

Rule Type: Message

Subsystem: DB2

Responds to DB2 IRLM lockouts caused by a system or DB2 failure within a shared data group.

DSNT501I

Rule Type: Message

Subsystem: DB2

Process DB2 Thread timeout conditions.

EOJXSYS

Rule Type: EOJ

Subsystem: z/OS

Demonstrates the logic that can be used to obtain status or event data from another system such as the maximum condition code (maxcc) of a batch job in order to perform some automated action on a local system.

EOMDEQ

Rule Type: End-of-memory

Subsystem: CA OPS/MVS

Dequeues remaining table editor and schedule manager enqueues after TSO users terminate.

EPITSO

Rule Type: Command

Subsystem: CA OPS/MVS

Demonstrates EPI capabilities by logging on a TSO user ID to get a data set name list based on the AOF rule set prefix and suffix.

GCM

Rule Type: Message

Subsystem: CA MIM

Keeps CA MIC internal messages out of OPSLOG.

IAT3100

Rule Type: Message

Subsystem: JES3

Informs the operator when JES3 has been initialized and issues a prompt to start the XYZ job.

IEA611I

Rule Type: MSG

Subsystem: z/OS

Process IEA611I dump messages and generate an alert with the captured dump dsn for abends that are monitored from the IEA794I sample rule.

IEA794I

Rule Type: MSG

Subsystem: z/OS

Process IEA794I dump messages and store relevant dump data in a unique CA OPS/MVS global variable so that it can be manipulated in the IEA611I sample rule which generates a detail email of the dump event.

IEA989I

Rule Type: MSG

Subsystem: z/OS

Restarts the SMSVSAM address space if it produces a specific abend. Demonstrates throttle logic to process this event on time in 10 minutes.

IEC606I

Rule Type: MSG

Subsystem: z/OS

Processes the first occurrence of a VTOC error condition from the specific DASD device within a sysplex. All other identical issuances of this failure for the same VTOC error will not be processed for the next 10 minutes. This prevents reacting to the same event more than once. Create dynamic JCL to invoke the ICKDSF utility to reinitialize the VTOC when the IEC606I event occurs.

IEE043I

Rule Type: Message

Subsystem: z/OS

Starts a sample external writer when the SYSLOG data set is full and informs the operator.

IEE362A

Rule Type: Message

Subsystem: z/OS

Submits a disk reader command to JES3 to read in the job when an SMF data set is full.

IEE391A

Rule Type: Message

Subsystem: z/OS

Starts a job that dumps a full SMF data set.

When z/OS issues message IEE391A, this rule issues the following command:

```
MVS START SMFDMP,DSN=smfdsn
```

No JCL or procedure for dumping the SMF data set is provided in the rule. You must tailor the rule to fit your needs.

Note: When the SMF data set name does not use the SYS1.MAN*n* naming convention, message IEE391A is issued instead of message IEE362A.

IEF

Rule Type: Message

Subsystem: z/OS

Issues information messages for job management messages IEF403I and IEF404I.

IEF176I

Rule Type: Message

Subsystem: z/OS

Stops the external writer when it becomes idle.

IEF433D

Rule Type: Message

Subsystem: z/OS

When a job wait is requested with the reply HOLD or NOHOLD, this rule issues an operator REPLY command using the reply number from the first word of the message.

IFB040I

Rule Type: Message

Subsystem: z/OS

When z/OS issues message IFB040I to indicate that the SYS1.LOGREC data set is full, this rule submits and issues an MVS START command and notifies four users. No JCL or procedure for dumping/printing or clearing the data set is provided in the rule. You must tailor the rule to fit your needs.

IKJ574I

Rule Type: Message

Subsystem: TSO

Reduces the number of Broadcast Data Set Full messages to one every five minutes.

IKT010D

Rule Type: Message

Subsystem: TSO

Replies to the message indicating that a STOP command was entered to stop TCAS, but a number of TSO users are still active. The sample replies with the FSTOP response. If you prefer, you can change the response to SIC.

IMSREGS

Rule Type: Message

Subsystem: IMS

Saves the name of an IMS region within an CA OPS/MVS global variable. The variable is needed within the aggressive system shutdown sample procedure SHUTSYS2 to initiate the proper shutdown of an IMS region that may have started outside of STATEMAN.

INFOSAMP

Rule Type: Message

Subsystem: z/OS

When z/OS issues message IEA404A for WTO buffers full, the rule executes a CLIST that inserts a record into the IBM INFO/SYSTEM problem database. This CLIST is included in the CA OPS/MVS sample CLIST library.

INVKISPF

Rule Type: Message

Subsystem: z/OS

Dispatches a program to invoke ISPF services.

IOS

Rule Type: Message

Subsystem: z/OS

Highlights all IOS messages.

IPLINFO

Rule Type: Message

Subsystem: z/OS

Obtains various IPL data information, stores that data in a variable, and then issues a MLWTO OPSLOG only message with the data.

The MLWTO can be used as a quick tool within OPSLOG to locate the started IPLs, and the variable will be accessed using the OPSINQRY sample OPS/REXX program as a tool to present the collected IPL data across MSF connected systems.

IPLSUPPR

Rule Type: Message

Subsystem: MVS

Aggressively suppresses the numerous WTOs generated during a system IPL.

Rule Type: MSG

Subsystem: z/OS

Issues commands after a system IPL has been performed.

IRA200I

Rule Type: Message

Subsystem: z/OS

Responds to the Auxiliary Storage Shortage message by adding a page data set and highlighting the message.

IRA201I

Rule Type: Message

Subsystem: z/OS

Responds to the Auxiliary Storage Shortage message by adding a page data set and highlighting the message.

JES2$TJ

Rule Type: Command

Subsystem: z/OS

Allows a user to issue a $TJ command to only a jobname that corresponds to the TSO user ID. For example, user ID TSOUSR1 can issue $TJ commands only to TSOUSR1x jobnames.

JOBINFO

Rule Type: CMD

Subsystem: z/OS

Create pseudo command rule to trigger JOBINFO OPS/REXX program to extract specific JES data for a job and display data back to the console.

MIMTAPE

Rule Type: TOD

Subsystem: N/A

Invokes the MIMTAPE OPS/REXX program to determine if any outstanding mount pendings are occurring for locally managed CA MIM tape devices.

MLWTO1

Rule Type: Message

Subsystem: z/OS

Demonstrate rule logic needed to manipulate data across different lines of an MLWTO message in order to take some automation action. Specific logic processes the IEA995I event.

MLWTO2

Rule Type: Message

Subsystem: z/OS

Demonstrate rule logic needed to store each data line of an MLWTO so it can be manipulated within an OPS/REXX program to perform some type of asynchronous automation. The MLWTO keyword of the)MSG specifier is utilized.

MLWTO3

Rule Type: Message

Subsystem: z/OS

Demonstrate rule logic needed to manipulate data across different lines of an MLWTO message in order to create a single alert message. Specific logic processes the DSNT376I event.

MLWTO4

Rule Type: Message

Subsystem: z/OS

Demonstrate rule logic needed to process MLWTO events without the MLWTO keyword being specified.

MSFROUTE

Rule Type: Command

Subsystem: z/OS

Enables operators to route console commands to a remote system using the MSF component of CA OPS/MVS.

MSFTERM

Rule Type: Global Variable

Subsystem: MSF

Deactivates all active MSF sessions and deletes all MSF-defined nodes when the rule is disabled during normal product termination.

MSGBHOLE

Rule Type: Message

Subsystem: z/OS

Demonstrates how to completely eliminate a message from the OPSLOG, SYSLOG, consoles, JESMSGLG (joblog), and JESYSMSG data sets.

OPAODIGL

Rule Type: Request

Subsystem: TSO

Provides a full screen control application for viewing or updating OPS/REXX global variables. This rule provides a good illustration of the power and flexibility of REXX programs.

Note: This program must be invoked under ISPF.

OPNFFNET

Rule Type: Command

Subsystem: NetView

Displays all copies of NetView in the system.

OPS1000J

Rule Type: Message

Subsystem: CA OPS/MVS

Prevents the messages that result from SAY statements in rules from appearing in the JES JOBLOG.

OPS3014O

Rule Type: Message

Subsystem: CA OPS/MVS

Sets SVCDUMP parameter to ON after ESTAE routine has set SVCDUMP to OFF and issued this message.

OPS3445O

Rule Type: Message

Subsystem: z/OS

Switches automatically to an alternate active OPSLOG when current live OPSLOG nears a wrap condition.

OPS3487O

Rule Type: Message

Subsystem: CA OPS/MVS

Detects when the CA OPS/MVS to CAICCI interface is activated and attempts to activate CCI connections.

OPS3716O

Rule Type: Message

Subsystem: CA OPS/MVS

Executes when the OSF execute processor detects that a server address space has been cancelled but not terminated.

OPS4402O

Rule Type: Message

Subsystem: z/OS

Triggers on the OPS/MVS message that is issued when some asid may be looping and generates a formal alert if a defined threshold is met. The MSGDRAINRATE and MSGTHRESHOLD parameters within CA OPS/MVS detect address spaces that issue excessive message traffic.

OPSAOF

Rule Type: Command

Subsystem: z/OS

Enables console operators to issue AOF commands and receive responses at the z/OS console.

OPSCMD

Rule Type: Security

Subsystem: CA OPS/MVS

Prohibits certain TSO user IDs from issuing the OPSCMD command. Replace the USERID parameter with a valid TSO user ID before you enable this rule.

OPSLGSCN

Rule Type: TOD

Subsystem: z/OS

Invokes OPSLGSCN OPS/REXX program to periodically scan OPSLOG looking for specific OPS/REXX compiler and execution error messages that might have occurred within rules and programs. Generates an alert email of these messages to a list of designated users.

OPSLOGCK

Rule Type: Time-of-day

Subsystem: CA OPS/MVS

Warns when OPSLOG sequence numbers are approaching a wrap condition.

OPSMEDS

Rule Type: Command

Subsystem: z/OS

Invokes Mainframe Environment Discovery Script (MEDS) for specific CA products.

OPSMSF

Rule Type: Command

Subsystem: CA OPS/MVS

Allows for display and control of MSF facilities from a console. It can also be used to start MSF on another system in a sysplex using the MVS ROUTE command.

OPSOF001

Rule Type: API

Subsystem: SOF

Demonstrates how to bring selected devices online in response to API event OPSOF001 issued by SOF.

OPSOSF

Rule Type: Command

Subsystem: CA OPS/MVS

Lets you display and control OSF servers from a console.

OPSPARM

Rule Type: Command

Subsystem: CA OPS/MVS

Lets you display and set CA OPS/MVS parameters from a console.

OPSTORE

Rule Type: Command

Subsystem: z/OS

Displays home address space virtual storage on the console.

PDT0170

Rule Type: MSG

Subsystem: z/OS

Processes data collector messages from CA Xmanager and invokes TSFBATCH Proc unload to submit detector data to TSF.

PLEX*, STAP*

Rule Type: Message

Subsystem: z/OS

Indicates various sample rules that provide support to sample SSM Sysplex applications.

REPLY

Rule Type: Command

Subsystem: JES3

Enables console operators to reply to WTOR messages by entering only a reply number and reply text. JES2 provides this facility as a standard feature. This rule provides the same capability for JES3.

SECAOF

Rule Type: Command

Subsystem: CA OPS/MVS

Allows console operators to enter AOF commands and receive the responses. This version executes a request rule in the server to prompt for a password for CA ACF2 validation.

SECAOFRQ

Rule Type: Request

Subsystem: z/OS

Works with the SECAOF command to CA ACF2 to protect a command.

SECOC

Rule Type: Security

Subsystem: z/OS

Illustrates how generalized resource rules can be used to protect the OPSCMD command processor.

SECREQ

Rule Type: Security

Subsystem: CA OPS/MVS

Allows all users to issue the OPSREQ TSO command.

SECSSM*

Rule Type: Security

Subsystem: n/a

Demonstrates techniques to secure SSM control from OPSVIEW panels.

SECWEBV*

Security

Subsystem: n/a

Demonstrates techniques to secure the OPSLOG WebView component.

SENTINEL

Rule Type: Time-of-day

Subsystem: N/A

Starts the QUERYRES REXX program every five minutes to query the status of resources and update global variables.

SHUT

Rule Type: Command

Subsystem: z/OS

Enables console operators to issue a shutdown for CICS. The rule starts the REXX program CICSSHUT in a CA OPS/MVS server address space to perform the shutdown.

SHUTSYS

Rule Type: Time-of-day

Subsystem: STATEMAN

This TOD rule is part of the aggressive system shutdown sample procedures (as described within the SHUTSYS1 OPS/REXX program sample). This rule issues highlighted MLWTO system shutdown status messages within a specified 10-minute shutdown window, and stops the security package and CA OPS/MVS if all resources have stopped. Manual intervention alerts are generated if resources are still active at the end of the ten-minute shutdown window.

SOFCMDR

Rule Type: Command

Subsystem: SOF

Secures access to SOF from MVS console commands by using resources defined to a security product.

SOFCMDU

Rule Type: Command

Subsystem: SOF

Secures access to SOF from MVS console commands issued by user IDs defined in the rule.

SOFSECR

Rule Type: Security

Subsystem: SOF

Secures access to SOF from OPS/REXX ADDRESS SOF and OPSVIEW panels by using resources defined to a security product.

SOFSECU

Rule Type: Security

Subsystem: SOF

Secures access to SOF from OPS/REXX ADDRESS SOF and OPSVIEW panels used by user IDS defined in the rule.

SPOOLMON

Rule Type: Message

Subsystem: JES2

Invokes sample SPOOLMON OPS/REXX program when JES2 TGS (spool space/track groups) threshold warnings are received. Application creates an alert of the problem which includes the jobname of the top spool user.

SSMALTSB

Rule Type: Request

Subsystem: n/a

Demonstrates using SSM v2 techniques to perform a reverse order shutdown of resources. Start A, then B, then C, but stop B, then A, then C.

SSMCNTL

Rule Type: Command

Subsystem: STATEMAN

Provides the ability to monitor and control SSM resources using a pseudo command that can be entered from anywhere a system command can be issued.

SSMEOJA

Rule Type: EOJ

Subsystem: z/os

Process all End-Of-Job events on a system if a large amount of batch jobs and STCs are being monitored in SSM and Type 30 records are being generated for STCs.

SSMEOJB

Rule Type: EOJ

Subsystem: z/os

Process all End-Of-Job events on a system if a large amount of batch jobs and STCs are being monitored in SSM and Type 30 records are not being generated for STCs.

SSMEOJC

Rule Type: EOJ

Subsystem: z/os

Process End-Of-Job events for a particular batch job or jobs that are being monitored within SSM.

SSMEXCPS

Rule Type: Request

Subsystem: N/A

An end-user TSO tool to display SSM mismatches (exceptions) from all MSF-connected systems.

SSMMOVE, SSMPLEXC

Rule Type: Request

Subsystem: N/A

Demonstrates using SSM v2 techniques to move resources between sysplex systems.

SSMXPREQ, SSMXSUBQ

Rule Type: Request

Subsystem: N/A

Demonstrates using SSM techniques to perform cross sysplex prerequisite and subrequisite checking.

STARTMII

Rule Type: Command

Subsystem: N/A

Intercepts a command and modifies command contents.

STCB4OPS

Rule Type: Message

Subsystem: N/A

Saves the names of all system related STCS that are started prior to CA OPS/MVS within global variable. The variable is needed within the aggressive system shutdown sample procedure SHUTSYS2 to bypass the stopping of these tasks that will be internally stopped by the operating system.

SUPPRALL

Rule Type: Message

Subsystem: n/a

Implements aggressive message suppression.

SYSINFO, SYSINFO2

Rule Type: Message, Request

Subsystem: N/A

Obtains various system-related IPL information for a system and saves it in a CA OPS/MVS shared variable database. Query the data from a TSO focal point.

SYSVALRT

Rule Type: Time-of-day

Subsystem: n/a

Invokes SYSVALRT OPS/REXX program to monitor and respond accordingly to any CA SYSVIEW Alerts.

SYSVE

Rule Type: Command

Subsystem: COMMAND

Issues CA SYSVIEW commands on the console with information returned to the console. This rule must be implemented together with the OPS/REXX program SYSVECMD.

TAPEMNT1, TAPEMNT2, TAPEMNT3, TAPEMNT4

Rule Type: Message, Delete-operator-message, Time-of-day

Subsystem: N/A

Creates notification alerts for outstanding tape mounts that are pending for some defined threshold.

TIMECHNG

Rule Type: Time-of-day

Subsystem: N/A

Schedules the TIMECHNG REXX EXEC to run in an OSF TSO server, prior to the time the system clocks are changed. This rule is typically used for Daylight Savings time changes. See the TIMECHNG sample REXX program.

TIMERUL2

Rule Type: Time-of-day

Subsystem: N/A

Illustrates a rule that executes at time intervals for multiple days. The example rule executes every 15 minutes starting on August 2, 2007, at 1:50 and continuing until December 31, 2007.

TIMERUL3

Rule Type: Time-of-day

Subsystem: N/A

Illustrates a rule that executes 10 seconds after it is enabled. The rule is disabled after its first and only execution.

TIMERULE

Rule Type: Time-of-day

Subsystem: N/A

Demonstrates how a rule can execute multiple times in one day. The example executes nine times between 3:00 and 4:00 on August 3, 2007.

TIMING*

Rule Type: Time-of-day

Subsystem: N/A

Illustrates comparative rules to be used as directed by CA OPS/MVS support.

TSOCHECK

Rule Type: Time-of-day

Subsystem: TSO

Monitoring time rule to trigger the TSOCHECK sample OPS/REXX program that monitors TSO/E access and response time.

TSS90001

Rule Type: Message

Subsystem: Security

Restarts the CA OPS/MVS security interface after CA Top Secret is active.

UEJM*

Rule Type: Message

Subsystem: z/OS, CA7

Demonstrates creating a focal point for UEJM messages destined to OPSLOG.

WAEE2OPS

Rule Type: GLV

Subsystem: z/OS

Interrogate and process job monitoring data stored in a global variable as set through CA Workload Automation EE script.

WAEE2SEC

Rule Type: SEC

Subsystem: z/OS

Allow CA Workload Automation EE to create a global variable through the WAEE2OPS sample CA WAEE script.

WTOBUF

Rule Type: Message

Subsystem: z/OS

Responds to IEA404A and IEA405E MCS console backlog messages by detecting and clearing the backed-up consoles.

WTORS

Rule Type: Command

Subsystem: z/OS

Creates pseudo command rule for manual display request of sysplex WTOR information.

XCOPSEOJ

Rule Type: EOJ

Subsystem: z/OS

Initiate CA XCOM failover procedures if CA XCOM abnormally terminates.

XCOPSMSG

Rule Type: MSG

Subsystem: z/OS

Set a flag to be used within XCOPSEOJ to indicate CA XCOM is requested to shutdown.

XCSA

Rule Type: OMEGAMON

Subsystem: z/OS

Performs a shutdown on CICS when a z/OS CSA shortage exists.

XJOBSTAT

Rule Type: CMD

Subsystem: z/OS

Processes a pseudo sysplex routed command that can be used to remotely determine if an ASID is active using the OPSTATUS().

ZEROAOF

Rule Type: Request

Subsystem: N/A

Extracts and retains any enabled AOF rule that has a zero fire count.

Appendix C: Sample OPS/REXX Programs

This section contains the following topics:

Supplied Sample OPS/REXX Programs

This section lists the details regarding the individual sample OPS/REXX programs as outlined in Appendix A, which overviews a specific automatable system component or event.

The following lists and describes all the supplied sample programs:

ABENDLOG

Program Type: OPS/REXX

Subsystem: z/OS

Processes abend data stored in a RDF table using the ABENDLOG AOF rule, and offloads to a sequential data set.

ADDRCA7

Program Type: OPS/REXX

Subsystem: z/OS

Demonstrates issuing and interrogating output from CA 7. The interface between CA OPS/MVS and CA 7 must be implemented as outlined in the *Administration Guide*.

ALLOCSPF

Program Type: REXX

Subsystem: ISPF

Allocates files necessary to run ISPF in a server.

AOFCMDS

Program Type: REXX

Subsystem: AOF

Issues a series of commands to ADDRESS AOF and retrieves the AOF command output. The AOF command output is sent to the user.

AOFINIT

Program Type: REXX

Subsystem: AOF

Demonstrates how to initialize the AOF, based on the environment.

ASID

Program Type: CLIST

Subsystem: z/OS

Finds the address space IDs for jobs executing on a machine. The job name is provided as input to the CLIST.

BATCHPRM

Program Type: JCL

Subsystem: IMS

Invokes the supplied OPIMTGCB REXX program as a batch version of the IDENTIFY IMS utility, which is OPSVIEW option 7.4.

CANWTR

Program Type: CLIST

Subsystem: z/OS

Cancels scheduled writer jobs.

catwto

Program Type: UNIX shell script

Subsystem: USS

Displays the UNIX shell script. The UNIX shell script is called by the log2wto shell script, which issues the catwto for each line in a log file.

Note: This script is only intended for use in conjunction with the log2wto script. The program name is in lowercase because the UNIX environment is case sensitive.

CGLOBALV

Program Type: C

Subsystem: ESI

Demonstrates how to use OPSLINK from a C/370 program. To use this sample, you must license the ESI component in advance.

CHECKTM5

Program Type: OPS/REXX

Subsystem: CA 1

Checks to see if CA 1 is active, and if it is, whether it is running TMS Version 5.2.

CHECKTMS

Program Type: OPS/REXX

Subsystem: CA 1

Checks to see if CA 1 is active.

CICS

Program Type: CLIST

Subsystem: CICS

Enables you to stop, start, or display status information about CICS regions.

CICSSHUT

Program Type: OPS/REXX

Subsystem: CICS

Performs a controlled CICS shutdown if possible and takes more drastic shutdown actions if necessary. The program expects to be invoked by the SHUT command rule from a z/OS console.

CLEARQ

Program Type: REXX

Subsystem: N/A

Purges the external data queue and returns to the caller.

CMDALL

Program Type: OPS/REXX

Subsystem: z/OS

Create a focal point of view for the command output of commands that are issued across many CA OPS/MVS MSF connected systems.

CMDAUTHW

Program Type: OPS/REXX

Subsystem: z/OS

Creates a verification WTOR back to the console that issued a specific command.

COBCMDS1

Program Type: COBOL

Subsystem: z/OS

Issues (z/OS, JES, VM, IMS, CICS) commands and returns the output to an array supplied by the caller.

COMPAOF

Program Type: REXX

Subsystem: AOF

Shows how a batch job can be used to compile an entire AOF rule set.

COMPREXX

Program Type: JCL

Subsystem: REXX

Demonstrates how to compile an OPS/REXX program from a batch job.

COMPRSDS

Program Type: OPS/REXX

Subsystem: z/OS

Demonstrates using the OPSUBMIT OPS/REXX function to submit JCL to compress a data set.

Note: This sample program is related to and scheduled by the APIPDSMN sample rule.

CPIND

Program Type: OPS/REXX

Subsystem: VM

Displays the result of a VM command.

DAPPL

Program Type: CLIST

Subsystem: VTAM

Displays currently active VTAM application IDs.

DATE2SEC

Program Type: REXX SUBROUTINE

Subsystem: N/A

Converts either a *yy/mm/dd hh:mm:ss* or *yyyymmdd hh:mm:ss* string value to the number of seconds past midnight on January 1, 1980.

DB2

Program Type: CLIST

Subsystem: DB2

Enables you to stop, start, cancel, and display active DB2 sessions.

DB2LOGDB

Program Type: OPS/REXX

Subsystem: DB2

Issue and store DB2 Display command output in a sequential file specifically a DIS DB(dbname) command.

DB2THRD

Program Type: OPS/REXX

Subsystem: DB2

Issue and manipulate the output of a DB2 Display command specifically a DIS THREAD(*) command.

DEFNVCON

Program Type: OPS/REXX

Subsystem: NetView

Defines CA OPS/MVS Subsystem Consoles to NetView.

DVTMNODE

Program Type: OPS/REXX

Subsystem: z/OS

Demonstrates the logic needed to issue commands to the system and collect and interrogate the command response, specifically the status of a VTAM node.

EMAILMSG

Program Type: REXX

Subsystem: JES

Create a focal application that can be called by any AOF rule or OPS/REXX program when a multi-line automated SMTP email needs to be generated.

EMAILTXT

Program Type: REXX

Subsystem: JES

Sends a single-line SMTP email.

EOJXSYS

Rule Type: OPS/REXX

Subsystem: z/OS

Demonstrates the logic that can be used to obtain status or event data from another system such as the maximum condition code (maxcc) of a batch job in order to perform some automated action on a local system.

EPICICS

Program Type: OPS/REXX

Subsystem: CICS

An EPI subroutine that executes CICS transactions and returns the results and response times.

EPITSO

Program Type: OPS/REXX

Subsystem: EPI

Demonstrates EPI capabilities by logging on a TSO user ID to get a data set name list based on the AOF rule set prefix and suffix.

GDITEST

Program Type: JCL

Subsystem: GDI

Shows how you can use the generic data set interface to route the output from a sequential file to the AOF for automation in messages rules.

GETSTORD

Program Type: OPS/REXX

Subsystem: z/OS

Examines the LDA (VSM local data area) and derives data concerning user-storage usage.

GTERMID

Program Type: REXX SUBROUTINE

Subsystem: z/OS

Gets the terminal ID associated with the current job.

IDMSAREA

Program Type: OPS/REXX

Subsystem: z/OS

Invoke and manipulate the collected DCMT D AREA command output. If the LOCK status, is in 'OFL' for any area then send out an alert message.

IEC606I

Program Type: OPS/REXX

Subsystem: z/OS

Creates JCL to invoke ICKDSF utility to reinitalize the VTOC when IEC606I events occur within a sysplex.

IN

Program Type: REXX SUBROUTINE

Subsystem: z/OS

Simulates an SAA REXX built-in function.

INITMGR

Program Type: OPS/REXX

Subsystem: JES2

Manages and controls some or all JES2 initiators based on day-of-week and time-of-day.

INSERT

Program Type: REXX

Subsystem: CA OPS/MVS

Demonstrates the ability of an OPS/REXX subroutine to simulate a built-in function of SAA REXX.

JOBINFO

Program Type: OPS/REXX

Subsystem: z/OS

Utilize the OPSJESX() to obtain JES SSI 80 function call data for a specific job and display the information to the requesting console.

LINKLIST

Program Type: OPS/REXX

Subsystem: z/OS

Displays the names of the data sets in the LNKLST.

log2wto

Program Type: UNIX shell script

Subsystem: USS

Displays UNIX shell script to route messages from a log file to z/OS CA Event Manager. The program name is in lowercase because the UNIX environment is case sensitive.

Example: log2wto *filename prefix*

filename is the name of the UNIX log file, and *prefix* is the optional character string that is inserted in front of each message in the log file. The prefix makes it possible for USS rules to determine the source of the message.

Program Type: OPS/REXX

Subsystem: z/OS

Causes tape volume dismounts to occur under CA MIA GLOBAL Tape Device Allocation Serialization for tape volumes that remain mounted in unallocated MIA-managed tape devices.

MEDSMIM

Program Type: OPS/REXX and CA OPS/MVS batch job

Subsystem: CA MIM

Gathers CA MIM complex environmental information quickly and ensures your CA MIM address spaces are setup and running optimally.

MIMTAPE

Program Type: OPS/REXX

Subsystem: CA MIM

Determines if any outstanding mount pendings are occurring for locally managed CA MIM tape devices.

MLWTOR

Program Type: OPS/REXX

Subsystem: z/OS

Uses the ADDRESS WTO host environment to demonstrate how to simulate the issuing of a multi-line WTOR.

MQQCHK

Program Type: OPS/REXX

Subsystem: CA OPS/MVS

Illustrates how to put and get messages on WebSphere MQ Series queues using the Address MQ host environment.

NETVIEW

Program Type: CLIST

Subsystem: NetView

Stops, starts, and displays NetView session status.

NOTIN

Program Type: REXX SUBROUTINE

Subsystem: z/OS

Simulates an SAA REXX built-in function.

OMVSPID

Program Type: OPS/REXX

Subsystem: USS

Illustrates how the D OMVS command can be used to determine or validate the OMVS process ID of an executing OMVS program. The PID is required by the UNIX kill command to stop a running process.

OP4UEXIT

Program Type: OPS/REXX

Subsystem: CA OPS/MVS

User command exit allows user to trigger any type of end-user automation against some SSM resource. Users can implement their own new line and primary command to be able to invoke a program. For example, an OPS/REXX validation program, SSM Note or a new ISPF application.

OPAMEPRM

Program Type: REXX

Subsystem: ISPF

Creates a dynamic parameter file for the AME report program.

OPAMINIT

Program Type: OPS/REXX

Subsystem: CA OPS/MVS

Provides support for the Automate ICLIST and IREXX parameters when modified.

OPCMDOUT

Program Type: REXX

Subsystem: CA OPS/MVS

Generates the OSCMD format of the CMDRESP(REXX) keyword output from the OPS/REXX format of the ADDRESS OPER command output.

OPCONCAT

Program Type: CLIST

Subsystem: OPSVIEW

Adds a data set to an existing concatenation of partitioned data sets.

Note: The OPCONCAT program is distributed in the OPS.CCLXCLS0 data set.

OPDECONC

Program Type: CLIST

Subsystem: OPSVIEW

Removes a data set from an existing concatenation of partitioned data sets.

Note: The OPDECONC program is distributed in the OPS.CCLXCLS0 data set.

OPINFSYS

Program Type: CLIST

Subsystem: INFO/SYS

Enables a record to be added to the IBM INFO/SYSTEM problem management database. Also includes a sample REXX rule that executes the CLIST when message IEA404A is issued.

OPSLGEXT

Program Type: OPS/REXX

Subsystem: z/OS

Utilize the OPSLOG() to extract filtered OPSLOG data.

OPRXINFO

Program Type: REXX

Subsystem: N/A

A working example of a REXX program that adds a problem record to the INFO/SYSTEM problem management database.

OPSFTP

Program Type: OPS/REXX

Subsystem: CA OPS/MVS

Copies one sequential data set or member to and from MVS host using FTP. The target node can be a PC with FTP server software (such as FileZilla Server) installed and configured.

OPSFTP supports both binary and text transfer modes and saves encrypted passwords in global variables.

OPSINFOT

Program Type: OPS/REXX

Subsystem: CA OPS/MVS

Demonstrates the use of all the OPSINFO function codes.

OPSLGSCN

Program Type: OPS/REXX

Subsystem: z/OS

ScanS OPSLOG looking for specific OPS/REXX compiler and execution error messages that might have occurred within rules and programs. Generates an alert email of these messages to a list of designated users.

OPSOF

Program Type: OPS/REXX

Subsystem: z/OS

Issues the SOF command supplied by the caller to the specified SOF server. Returns the command response to the console specified by the caller.

Note: This sample program is related to and invoked by the OPSOF sample rule.

OPTNGCOL

Program Type: OPS/REXX

Subsystem: CA OPS/MVS

Adds TNGNOTIFY and RESOURCE_TXT columns to all System State Manager resource tables and to the directory table.

OPVMGLV

Program Type: REXX

Subsystem: Server

Extracts client messages from GLV storage and writes them as individual records to an output DASD file.

OPVMJCL

Program Type: JCL

Subsystem: Server

Shows a JCL sequence which can be used to activate the z/OS server.

OVERLAY

Program Type: REXX SUBROUTINE

Subsystem: z/OS

Simulates an SAA REXX built-in function.

PAZGEVENT

Program Type: OPS/REXX

Subsystem: z/OS

Generate a CA Process Automation event from CA OPS/MVS.

PAZSPROC

Program Type: OPS/REXX

Subsystem: z/OS

Start a CA Process Automation process from CA OPS/MVS.

PAZSSM

Program Type: OPS/REXX

Subsystem: z/OS

Demonstrate the extension of CA OPS/MVS SSM functionality to the distributed environment by controlling SAP, running on Linux, through CA Process automation. The functions of these sample applications are invoked using SSM actions as specified in the action table.

PAZWSCMD

Program Type: OPS/REXX

Subsystem: z/OS

Create the common portion of a web services client request to be sent to CA Process Automation using Address USS.

PLEXSSM

Program Type: REXX

Subsystem: CA OPS/MVS

Lets sysplex and shared tape (STAPE) resources be monitored using sample message rules that update the resource status based on the severity of messages issued in the sysplex. The status of a resource can be NORMAL, WARNING, or CRITICAL. If the current state of a resource is UNKNOWN, then the resource is most likely not in use. STAPE resources are individual tape devices that have been defined as auto-switchable. Parsing the response to an operator DISPLAY command, which displays all auto-switchable devices, automatically populates the STAPE resource table. Use OPSVIEW option 4.11.2 to view these resources.

The monitored sysplex resources include the following:

- Automatic restart manager (ARM)
- Coupling facility resource manager (CFRM)
- Cross-system coupling facility (XCF)
- Cross-system extended services (XES)
- Sysplex timers (ETR)
- Sysplex failure management (SFM)
- System logger (LOGR)
- Workload manager (WLM)

Important! The PLEXSSM application, built on System State Manager (SSM), monitors sysplex resources and does not let actions be taken against a resource, such as starting, stopping, or recovering a resource associated with an application. Such actions are accomplished using customer extensions to the application.

For installation instructions, see the section Installation and Configuration Considerations for PLEXSSM.

PLICMDS1

Program Type: PL/1

Subsystem: z/OS

Issues system (z/OS, JES, VM, IMS, CICS) commands and returns the output to an array supplied by the caller.

PLICMDS2

Program Type: PL/1

Subsystem: TSO

Issues TSO commands and returns the output to an array supplied by the caller.

PLIGLOBV

Program Type: PL/1

Subsystem: CA OPS/MVS

Uses global variables from a high-level language program.

PRINTMSG

Program Type: CLIST

Subsystem: TSO

Prints a message on a printer.

PRODPERF

Program Type: OPS/REXX

Subsystem: CA OPS/MVS

Resets the CA OPS/MVS execution statistics in the PRODPERFORM parameter group.

PROFNOTE

Program Type: REXX

Subsystem: PROFS

Prepares, generates, and sends a PROFS note.

QUERYRES

Program Type: REXX

Subsystem: z/OS

Queries the status of predefined resources and stores the information in global variables.

RDFSIZE

Program Type: OPS/REXX

Subsystem: z/OS

Calculates the amount of storage that Automate relational tables will require in CA OPS/MVS.

RESETOSF

Program Type: OPS/REXX

Subsystem: z/OS

End-User OPS/REXX pgm to stop or force CA OPS/MVS servers and reset (delete) all server requests in the specified server queue.

REXXIMAC

Program Type: CLIST

Subsystem: ISPF

This is an ISPF edit macro that defines OI and OX as edit macro programs.

REXXIN

Program Type: OPS/REXX

Subsystem: z/OS

Reads a file in REXX.

REXXOUT

Program Type: REXX

Subsystem: z/OS

Writes a file in REXX.

RXGRAPHIC

Program Type: OPS/REXX

Subsystem: ISPF

Displays a graphic image in REXX.

SAMPTSO

Program Type: OPS/REXX

Subsystem: TSO

Provides a full-screen control application for issuing TSO commands and allows paging in the output.

SCANSSCT

Program Type: OPS/REXX

Subsystem: z/OS

Displays the names of all subsystems in use.

SEC2DATE

Program Type: REXX SUBROUTINE

Subsystem: N/A

Converts a seconds value since midnight January 1, 1980 into a 17-byte date and time string in the format *yyyymmdd hh:mm:ss*.

SENDPROF

Program Type: OPS/REXX

Subsystem: PROFS

Prepares, generates, and sends a PROFS note.

SETSLIP

Program Type: OPS/REXX

Subsystem: z/OS

Sets a dynamic SLIP trap. CA Customer Support personnel may direct you to use this sample.

SHUTCICS

Program Type: OPS/REXX

Subsystem: z/OS

Initiates shutdown for a CICS region.

SHUTMUFT

Program Type: OPS/REXX

Subsystem: z/OS

Stops a CA Datacom region. This sample program contains examples that use the OPSJES2 function to interrogate a specific initiator class, the OPSTATUS function to evaluate specific active jobs, and an example of creating a dynamic TOD rule.

SHUTSYS1

Program Type: OPS/REXX

Subsystem: z/OS

Initiates STATEMAN system shutdown procedures. This sample and other sample AOF rules and OPS/REXX programs work together to create a system shutdown application. Specific details are outlined within this SHUTSYS1 sample.

SHUTSYS2

Program Type: OPS/REXX

Subsystem: z/OS

Triggered using the SHUTSYS1 sample program. This sample stops non SSM-related tasks, and then initiates the STATEMAN shutdown. This sample and other sample AOF rules and OPS/REXX programs work together to create a system shutdown application. Specific details are outlined within the SHUTSYS1 sample.

SHUTZFS

Program Type: OPS/REXX

Subsystem: z/OS

Initiates shutdown for the OMVS zFS file system.

STARTZFS

Program Type: OPS/REXX

Subsystem: z/OS

Restarts the OMVS zFS file system after the following command was issued:

```
F OMVS,STOPPFS=ZFS
```

SPOOLMON

Program Type: OPS/REXX

Subsystem: JES2

Uses the CA OPS/MVS to CA SYSVIEW interface to monitor and respond to JES2 TGS (spool space/track groups) threshold warnings. Application creates an alert of the problem, which includes the job name of the top spool user.

SSMGLSST

Program Type: OPS/REXX

Subsystem: STATEMAN

Handles the BEGIN, ADD, UPDATE, and DELETE global events of SSM. In this sample, a new SSM-related table of resources whose statuses must be transmitted to other systems is built and maintained. The SSM global event table invokes this program through the SSMGEVNT request rule.

SSMMXREQ

Program Type: OPS/REXX

Subsystem: STATEMAN

Handles the XPREREQ, XSUBREQ, MPREREQ, MSUBREQ, and MATCH process events of SSM. A local SSM auxiliary table is used by these sample exits to evaluate cross-system prerequisite status requests. The SSM global event sample program SSMGLSST maintains the auxiliary table.

SSMPSREQ

Program Type: OPS/REXX

Subsystem: STATEMAN

The logic of this program emulates the standard prerequisite and subrequisite logic of the SSM engine. It can be modified and called from any REXX program to emulate a prerequisite and subrequisite evaluation.

SSMQUERY

Program Type: OPS/REXX

Subsystem: z/OS

Locate the SSM resource table in which a particular resource resides.

SSMSNSST

Program Type: OPS/REXX

Subsystem: STATEMAN

Transmits the SSM resource status records built by the sample SSM global event program, SSMGLSST, to the remote status SSM auxiliary table on the target systems. The SSM XPREREQ and XSUBREQ process events use the remote status table on each system to evaluate the status of cross-system prerequisites and subrequisites.

SSMTREE

Program Type: OPS/REXX

Subsystem: CA OPS/MVS

Generates a formatted file representation of a system state managed resource table. You can use this file with GRAPHVIZ to display a graphical representation of the table's resource relationships.

SSMUSAPL

Program Type: OPS/REXX

Subsystem: STATEMAN

Determines the current state of a USS application-level pseudo resource using the states of the component SSM USS resources and synchronizes the desired states of the component resources. This program is called by the SSMUSSRQ request rule. For an explanation of the use of a USS application resource, see the documentation in the beginning of the program.

SSMUSRES

Program Type: OPS/REXX

Subsystem: STATEMAN

Determines the current state of a USS resource using the OPSUSS REXX function and starts and stops a USS resource. This program is called by the SSMUSSRQ request rule. For the expected RDF table columns required to use this program, see the documentation in the beginning of the program.

Program Type: OPS/REXX

Subsystem: z/OS

Demonstrates a coding technique that can be used to monitor and control USS daemon server processes such as INETD within System State Manager (SSM).

SSMXCHCK

Program Type: OPS/REXX

Subsystem: z/OS

Determines the state of a remotely monitored SSM resource within a MSF connected or sysplex environment.

SSM2XCEL

Program Type: OPS/REXX

Subsystem: z/OS

Insert SSM resource and action table data into a sequential dsn so that it can be FTP'd and viewed as an Xcel document.

STARTIMS

Program Type: CLIST

Subsystem: IMS

Performs a typical IMS startup.

STOPUSS

Program Type: OPS/REXX

Subsystem: USS

Demonstrates how to stop all the OSF USS servers. You may need to use this procedure if you plan to shut down the Job Entry Subsystem (JES) prior to shutting down CA OPS/MVS.

SVCTABLE

Program Type: OPS/REXX

Subsystem: z/OS

Displays the system SVC table.

SYMBOLS

Program Type: OPS/REXX

Subsystem: z/OS

Displays the static system symbols.

Program Type: OPS/REXX

Subsystem: z/OS

Demonstrate a coding technique that can be used to obtain and interrogate specific CA SYSVIEW related data. Specifically the coupling facility related data, MQ related data, Page data set related data, and WLM related data.

SYSVCICS

Program Type: OPS/REXX

Subsystem: z/OS

Outlines and demonstrates the process of creating an effective CA OPS/MVS application that monitors and responds to CICS threshold alerts collected by CA SYSVIEW. This application provides a foundation of implementing a more granular automated decision making application, that is needed when processing these CA SYSVIEW alerts.

SYSVALRT

Program Type: OPS/REXX

Subsystem: z/OS

Create proactive automation between the CA OPS/MVS and CA SYSVIEW interface that alerts on and resolves potential problem ASIDs before they impact system performance.

SYSVCHCK

Program Type: OPS/REXX

Subsystem: z/OS

Demonstrates a coding technique that you can use to obtain and interrogate various CA SYSVIEW related data and generate SMTP email alerts for any exceeded defined thresholds.

SYSVCTDQ

Program Type: OPS/REXX

Subsystem: z/OS

Invoke the CA SYSVIEW CICS CTDATA GLOBAL command to obtain transient data queues for all active CICS regions and generate an alert for any TDQ that that has a current QCount greater than a defined threshold.

SYSVDSNX

Program Type: OPS/REXX

Subsystem: z/OS

The logic within this OPS/REXX program demonstrates the basic code needed to extract data set extent usage information for data sets active to a specific job. The CA SYSVIEW interface will be utilized to obtain this information.

SYSVECMD

Program Type: OPS/REXX

Subsystem: CA OPS/MVS

Issues CA SYSVIEW commands. This program is designed to run under a CA OPS/MVS server; therefore, it must be placed in the SYSEXEC (or equivalent) concatenation of an OSF TSO class server.

SYSVINIT

Program Type: OPS/REXX

Subsystem: z/OS

Determine if a batch jobs execution or clocktime is exceeding defined run times based on the initiator class in which it is running.

SYSVMSU

Program Type: OPS/REXX

Subsystem: z/OS

Process MSU4HAVG threshold alerts (Averages MSU usage). For warning/problem alerts, output from the TOPCPU CA Sysview command will be obtained and emailed through SMTP to a specific list of email IDs. An 'All OK' email will be generated when the MSU4HAVG alert returns a 'NORMAL' status.

TCPIPUSS

Program Type: OPS/REXX

Subsystem: z/OS

Demonstrates a technique used to query the statuses of z/OS UNIX processes associated to a particular ASID.

TIMECHNG

Program Type: OPS/REXX

Subsystem: CA OPS/MVS

Waits for the time change to complete and then disables and reenables all non-dynamic TOD rules. See the related TIMECHNG sample rule.

TSOAUTH

Program Type: OPS/REXX

Subsystem: TSO

Displays the TSO attributes for a TSO user or server address space. Useful for debugging TSO-related problems.

TSOCHECK

Program Type: OPS/REXX

Subsystem: TSO

Sample program that uses the CA OPS/MVS External Program Interface (EPI) to respond to TSO/E logon or poor response time failures.

UCC7

Program Type: REXX

Subsystem: CA 7

Demands a job from the scheduler.

WORDFIND

Program Type: REXX

Subsystem: ISPF

Extracts all lines of a file containing a given string.

WTOBUF

Program Type: OPS/REXX

Subsystem: z/OS

Responds to IEA404A and IEA405E MCS console backlog messages by detecting and clearing the backed up consoles.

XSYSASID

Program Type: OPS/REXX

Subsystem: z/OS

Determine if an ASID is active on a remote CA OPS/MVS MSF connected system, or within a specific sysplex system.

Installation and Configuration Considerations for PLEXSSM

Following are the steps required to install and configure the sysplex resource monitor for the PLEXSSM sample program:

1. Start CA OPS/MVS, and then run the PLEXSSM REXX EXEC to build the following RDF tables:
   - PLEXRTBL-sysplex resource table
   - PLEXATBL-sysplex resource action table
   - STAPRTBL-STAPE resource table
   - STAPATBL-STAPE resource action table
2. Copy the PLEX* and STAP* rules to the active rule set and set the auto-enable option:
   - SSMBEGIN-Includes modifications to set the DESIRED state of sysplex resources
   - PLEXINIT-Rule that executes to set the current state when sysplex resources are in the UNKNOWN state
   - PLEX*ARM-Rules to maintain the status of ARM resources
   - PLEX*ASW-Rules to maintain the status of auto-switch resources
   - PLEXCFRM-Rules to maintain the status of CFRM resources
   - PLEXETR-Rules to maintain the status of ETR resources
   - PLEXGRS-Rules to maintain the status of GRS resources
   - PLEX*LOG-Rules to maintain the status of LOGGER resources
   - PLEXSFM- Rules to maintain the status of SFM resources
   - PLEXWLM-Rules to maintain the status of WLM resources
   - PLEX*XCF-Rules to maintain the status of XCF resources
   - PLEX*XES-Rules to maintain the status of XES resources
   - STAPINIT-Rules that execute to populate the STAPRTBL and STAPATBL tables and set the current state when STAPE resources are in the UNKNOWN state
   - STAP*ASW-Rules to maintain the status of the individual STAPE devices

Appendix D: CA OPS/MVS Health Checks

This Appendix describes health checks for C*A OPS/MVS*. The product owner for all CA OPS/MVS health checks is CA_OPSMVS.

This section contains the following topics:

OPSMVS_ALLOC_OPSLOG

This health check checks to see if you have allocated a data set for OPSLOG use. If it detects a possible loss of historical information, a low-severity Health Check exception is reported and CA OPS/MVS continues normally. This condition is checked periodically.

Best Practice

Make sure you allocate a data set for OPSLOG use. Otherwise, historical information will most certainly be lost. You can activate the OPSLOG facility without allocating a data set, but in that case, log information is kept in storage and becomes subject to periodic overlays as the storage fills up.

Parameters Accepted

No

Debug Support

No

Verbose Support

No

Reference

For information on allocating an OPSLOG data set, see the DEFDIV utility and the OPSSPA00 OPS/REXX program. This is further documented in the section Define OPSLOG and Checkpoint VSAM Linear Data Sets in the *Installation Guide*.

Messages

See the *Messages Reference* and the chapter "Migration and Upgrade Considerations" in the *Installation Guide.*

OPSMVS_ALLOC_SYSCHK1

This health check checks to see if the SYSCHK1 data set has been allocated. This data set maintains high-impact system components, such as global variables and RDF tables. Critical system automation will be compromised if these resources are not available. If the SYSCHK1 data set is not allocated, a medium severity health check exception is reported and CA OPS/MVS continues. This condition is checked once.

Best Practice

Make sure you allocate a SYSCHCK1 data set.

CA OPS/MVS can run at a significantly reduced level without a SYSCHK1 data set, but functionality is severely impaired.

Parameters Accepted

No

Debug Support

No

Verbose Support

No

Reference

For information on allocating an SYSCHK1 data set, see the DEFDIV utility and the OPSSPA00 OPS/REXX program. This is further documented in the section Define OPSLOG and Checkpoint VSAM Linear Data Sets in the *Installation Guide*.

Messages

See the *Messages Reference* and the chapter "Migration and Upgrade Considerations" in the *Installation Guide.*

OPSMVS_PARM_AOFHLQ

This health check checks to see if the high level qualifier for the rule data set name is ‘SYS1’. This condition can cause potentially high overhead. This condition is checked once, and reports a medium-severity health check exception if the HLQ ‘SYS1’ is detected.

Best Practice

Specify a less common high-level qualifier for the value of the RULEPREFIX parameter that is consistent with your data set naming standards. Avoid the familiar system prefix ‘SYS1’.

Parameters Accepted

No

Debug Support

No

Verbose Support

No

Reference

For more information, see the section Setting a Prefix and Suffix for Rule Sets in the *Installation Guide*.

For more information on the RULEPREFIX command, see the *Command and Function Reference*.

Messages

See the *Messages Reference* and the chapter "Migration and Upgrade Considerations" in the *Installation Guide.*

OPSMVS_PARM_AOFMAX

This health check measures the REXX external data queue's (EDQ) use of AOF rules against a threshold percentage. AOF rules are intended for maximum performance and small amounts of output. A rule that contributes to excessive use of EDQ may result in loss of data and performance degradation.

This condition is checked periodically, and reports a medium severity health check exception if the threshold is exceeded.

Best Practice

Review and improve CA OPS/MVS automation applications that are queuing a high number of entries in any of the external data queues. If your applications need additional EDQ capacity, increase the value of the AOFMAXQUEUE parameter by a reasonable amount.

Parameters Accepted

THRESHOLD(*nnn*)

Specifies the threshold queue percentage. *nnn* is a number between 0 and 100, inclusive.

Default: 80

Debug Support

No

Verbose Support

No

Reference

See the *AOF Rules Guide* and the discussion of the AOFEDQHIGH and AOFEDQWARNTHRESH parameters in the *Parameter Reference*.

Messages

See the *Messages Reference* and the chapter "Migration and Upgrade Considerations" in the *Installation Guide.*

OPSMVS_PARM_CMDMAX

This health check samples the rate of operator commands issued per second by any automation application. The percentage of the rate is calculated against the COMMANDMAX parameter, and is measured against the threshold value provided as the parameter *nn*. Exceeding the COMMANDMAX value can cause a CA OPS/MVS shutdown, so this health check can contribute to system up-time.

This condition is checked periodically, and reports a medium-severity health check exception if the threshold is exceeded.

Best Practice

Review and improve CA OPS/MVS automation applications that are causing a high number of operator commands to be issued. If nothing can be addressed in the short-term, increase the COMMANDMAX parameter value.

You can also eliminate this health check exception by increasing the value of the COMMANDRATE parameter.

Parameters Accepted

THRESHOLD(*nnn*)

Specifies a threshold rate percentage. *nnn* is a number between 0 and 100, inclusive.

Default: 80

Debug Support

No

Verbose Support

No

Reference

See the *Parameter Reference*.

Messages

See the *Messages Reference* and the chapter "Migration and Upgrade Considerations" in the *Installation Guide.*

OPSMVS_PARM_MSGMAX

This health check samples the rate of messages issued per second. The rate is calculated as a percentage of the MESSAGEMAX parameter, and compared against the threshold value provided as the parameter nn. Exceeding the MESSAGEMAX value can cause a system shutdown, so this health check can contribute to system up-time.

This condition is checked periodically, and will report a medium severity health check exception if the threshold is exceeded.

Best Practice

Review and improve CA OPS/MVS automation applications that issue a high number of operator messages. If nothing can be addressed in the short-term, increasing the value of MESSAGEMAX parameter value. You can also eliminate this health check exception by increasing the value of MESSAGERATE parameter.

Parameters Accepted

THRESHOLD(*nnn*)

Specifies a threshold rate percentage. *nnn* is a number between 0 and 100, inclusive.

Default: 80

Debug Support

No

Verbose Support

No

Reference

See the *Parameter Reference*, and the section Using Message Control Parameters in the *Administration Guide.*

Messages

See the *Messages Reference* and the chapter "Migration and Upgrade Considerations" in the *Installation Guide.*

OPSMVS_PARM_PROCBLK

This health check ensures that enough process blocks are available for CA OPS/MVS events or requests.

This is accomplished through the following process:

1. The PROCESS parameter determines how many process blocks are allocated in the extended private area of the CA OPS/MVS main address space when the CA OPS/MVS address space initializes.
2. The value of the SSEXEXITHICOUNT parameter indicates the high-water mark for the number of used process blocks.
3. The percentage of used process blocks is calculated against the PROCESS parameter, which is then measured against the threshold value provided as the parameter.
4. When an attempt to allocate a process block from the process block pool fails because no process blocks were available, the check provides the number of failed attempts and the date and time of the last failed allocation attempt.

This condition is checked periodically, and reports a medium-severity health check exception if the threshold is exceeded.

Best Practice

Allocating the right number of process blocks is critical. The number cannot be too low, because each event processed by CA OPS/MVS requires its own process block. If a process block is not available, then CA OPS/MVS will not capture or respond to the event, which in turn could lead to undesirable results on your system. Furthermore, setting the value too high has its own implications; the number of process blocks you specify may use so much virtual storage that CA OPS/MVS fails to function correctly.

Parameters Accepted

THRESHOLD(*nnn*)

Specifies a threshold rate percentage. *nnn* is a number between 0 and 100, inclusive.

Default: 80

AGE(*nn*)

Specifies delay between first exception and message informing that problem has not been fixed. *nn* is a number of hours between 0 and 24, inclusive.

Default: 6

Debug Support

No

Verbose Support

No

Reference

See the *Parameter Reference*.

Messages

See the *Messages Reference* and the chapter "Migration and Upgrade Considerations" in the *Installation Guide.*

OPSMVS_TSOMAXQUSG

This health check measures the maximum OSF TSO queue utilization. The percentage of maximum number of commands that have been on the OSF TSO queue is calculated against the OSFQUE parameter representing the maximum number of commands that the OSF TSO queue can hold. The percentage is measured against the threshold value provided as the parameter.

This condition is checked periodically, and reports a medium-severity health check exception if the threshold is exceeded.

Best Practice

This queue is where CA OPS/MVS sends TSO commands to be executed in the OSF TSO servers. The OSF execution scheduler dispatches these commands to OSF TSO servers as the servers become available to process work. The key objective of this health check is to provide you with appropriate warnings to prevent the situation where this queue overflows and TSO commands are never executed.

Parameters Accepted

THRESHOLD(*nnn*)

Specifies a threshold rate percentage. *nnn* is a number between 0 and 100, inclusive.

Default: 80

Debug Support

No

Verbose Support

No

Reference

See the *Parameter Reference*, *Administration Guide*, and *Command and Function Reference*.

Messages

See the *Messages Reference* and the chapter "Migration and Upgrade Considerations" in the *Installation Guide.*

OPSMVS_TSPMAXQUSG

This health check measures the maximum OSF TSP queue utilization. The percentage of maximum number of commands that have been on the OSF TSP queue is calculated against the OSFTSPQUE parameter representing the maximum number of commands that the OSF TSP queue can hold. The percentage is measured against the threshold value provided as the parameter.

This condition is checked periodically, and reports a medium-severity health check exception if the threshold is exceeded.

Best Practice

This queue is where CA OPS/MVS sends TSO commands to be executed in the OSF TSP servers. The OSF execution scheduler dispatches these commands to OSF TSP servers as the servers become available to process work. The key objective of this health check is to provide you with appropriate warnings to prevent the situation where this queue overflows and TSO commands are never executed.

Parameters Accepted

THRESHOLD(*nnn*)

Specifies a threshold rate percentage. *nnn* is a number between 0 and 100, inclusive.

Default: 80

Debug Support

No

Verbose Support

No

Reference

See the *Parameter Reference*, *Administration Guide*, and *Command and Function Reference*.

Messages

See the *Messages Reference* and the chapter "Migration and Upgrade Considerations" in the *Installation Guide.*

OPSMVS_TSLMAXQUSG

This health check measures the maximum OSF TSL queue utilization. The percentage of maximum number of commands that have been on the OSF TSL queue is calculated against the OSFTSLQUE parameter representing maximum number of commands that the OSF TSL queue can hold. The percentage is measured against the threshold value provided as the parameter.

This condition is checked periodically, and reports a medium-severity health check exception if the threshold is exceeded.

Best Practice

This queue is where CA OPS/MVS sends TSO commands to be executed in the OSF TSL servers. The OSF execution scheduler dispatches these commands to OSF TSL servers as the servers become available to process work. The key objective of this health check is to provide you with appropriate warnings to prevent the situation where this queue overflows and TSO commands are never executed.

Parameters Accepted

THRESHOLD(*nnn*)

Specifies a threshold rate percentage. *nnn* is a number between 0 and 100, inclusive.

Default: 80

Debug Support

No

Verbose Support

No

Reference

See the *Parameter Reference*, *Administration Guide*, and *Command and Function Reference*.

Messages

See the *Messages Reference* and the chapter "Migration and Upgrade Considerations" in the *Installation Guide.*

OPSMVS_USSMAXQUSG

This health check measures the maximum USS queue utilization. The percentage of maximum number of commands that have been on the USS queue is calculated against the USSQUE parameter representing maximum number of commands that the USS queue can hold. The percentage is measured against the threshold value provided as the parameter.

This condition is checked periodically, and reports a medium-severity health check exception if the threshold is exceeded.

Best Practice

This queue is where CA OPS/MVS sends TSO commands to be executed in the USS servers. The OSF execution scheduler dispatches these commands to USS servers as the servers become available to process work. The key objective of this health check is to provide you with appropriate warnings to prevent the situation where this queue overflows and TSO commands are never executed.

Parameters Accepted

THRESHOLD(*nnn*)

Specifies a threshold rate percentage. *nnn* is a number between 0 and 100, inclusive.

Default: 80

Debug Support

No

Verbose Support

No

Reference

See the *Parameter Reference*, *Administration Guide*, and *Command and Function Reference*.

Messages

See the *Messages Reference* and the chapter "Migration and Upgrade Considerations" in the *Installation Guide.*

OPSMVS_OPJ2CB

This health check checks whether your product JES2 offset table corresponds to JES2 on this system. It detects the following three possibilities:

- The CA OPS/MVS JES2 offset table module, OPJ2CB, has been assembled and linked with the current version of JES2.
- A default JES2 offset table is being used because the module OPJ2CB has not been assembled using the current release of JES2 on this system.
- A JES2 offset table for the current release of JES2 cannot be found and automation that relies on the OPSJES2 REXX function will not function correctly.

This condition is checked once, and reports a medium-severity health check exception if the threshold is exceeded. This health check will not be activated in a JES3 environment.

Best Practice

An accurate JES2 offset table ensures that data returned by the OPSJES2 REXX function is completely reliable.

Parameters Accepted

No

Debug Support

No

Verbose Support

No

Reference

See the *Installation Guide* and the *Command and Function Reference*.

Messages

See the *Messages Reference* and the chapter "Migration and Upgrade Considerations" in the *Installation Guide.*

Index

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X