Unicenter® CA-OPS/MVS®

Unicenter CA-OPS/MVS Event Management and Automation: Getting Started

Student Guide

PV001

Computer Associates®

ED0537X1GSE

One Computer Associates Plaza, Islandia, NY 11788-7000

Call Computer Associates technical services for any information not covered in this manual or the related publications. In North America, see your Computer Associates *Product Support Directory* for the appropriate telephone number to call for direct support, or you may call 1-800-645-3042 or 631-342-4683 and your call will be returned as soon as possible.

Outside North America, contact your local Computer Associates technical support center for assistance.

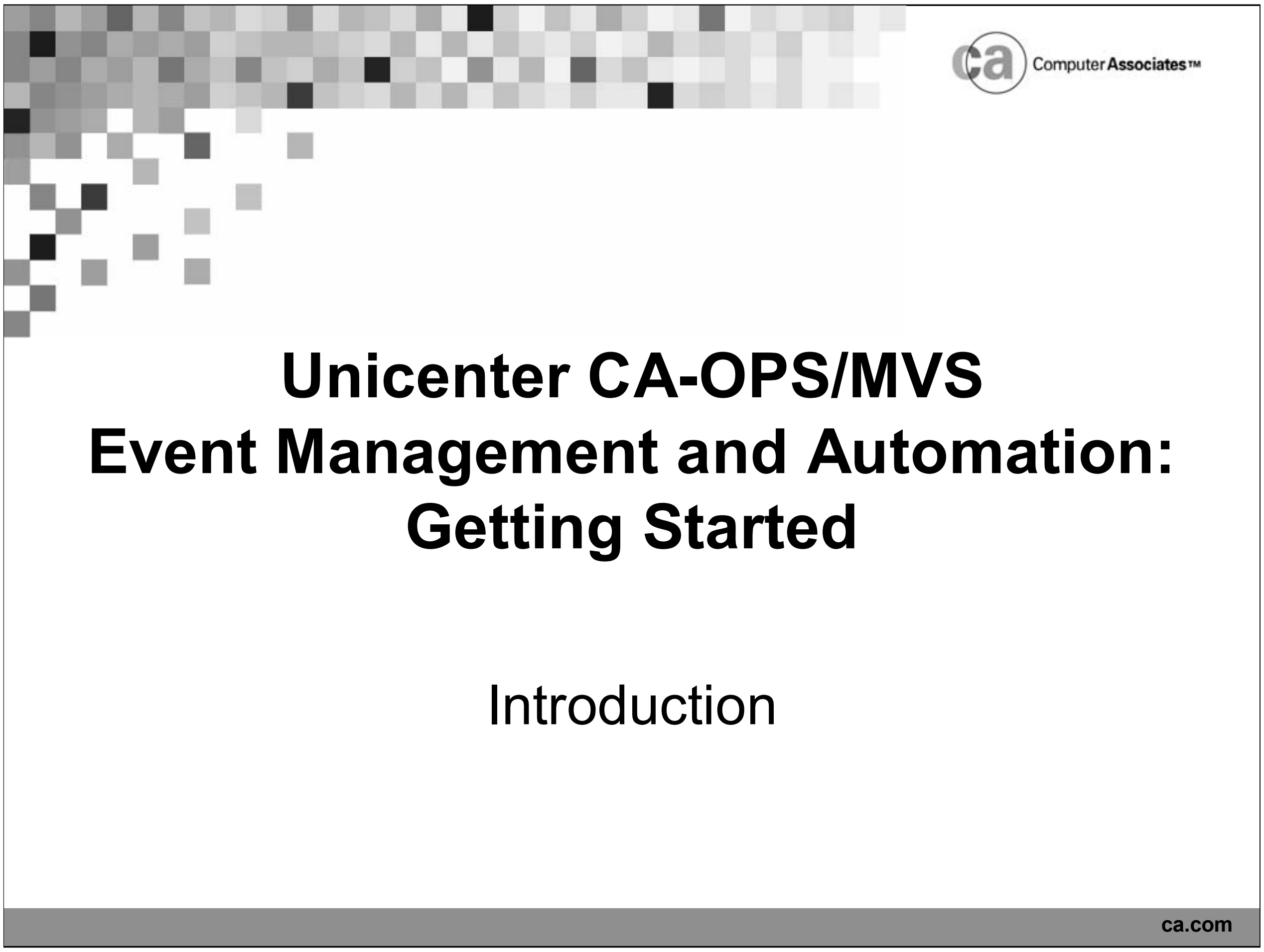
▪ Table of Contents

7 REXX Basics

8 AOF Variables

▪ Table of Contents

Notes:

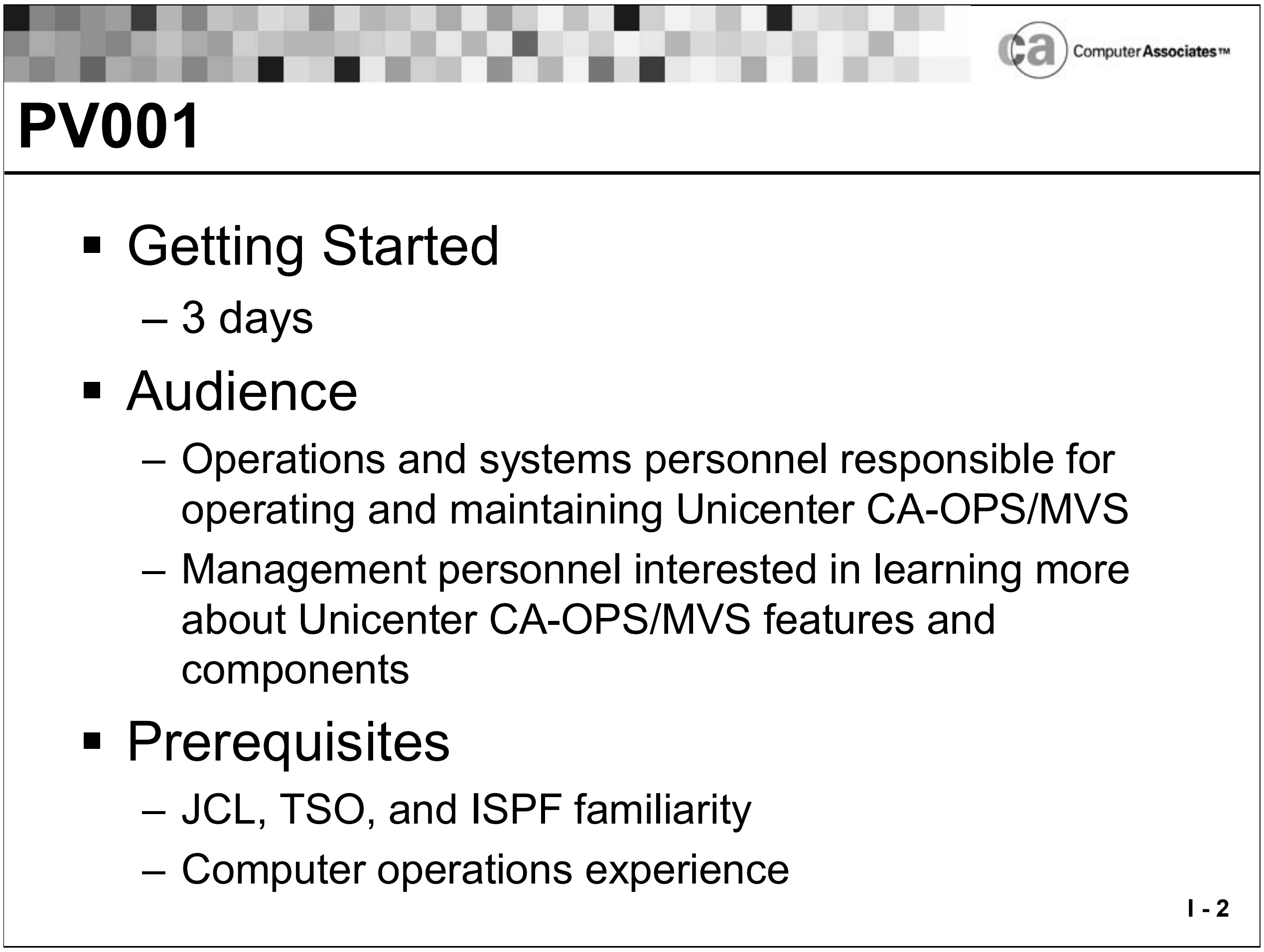


Computer Associates™

Unicenter CA-OPS/MVS Event Management and Automation: Getting Started

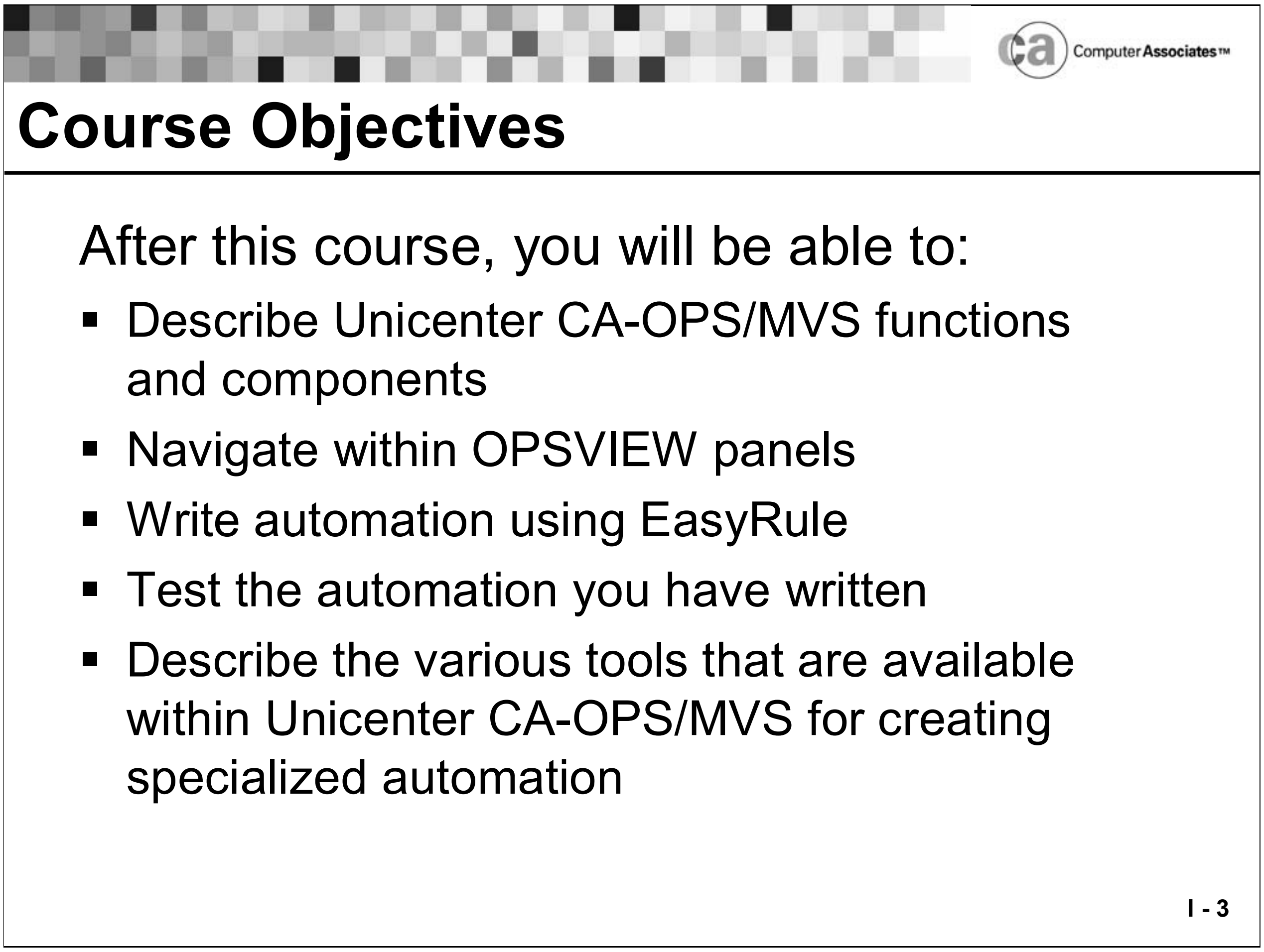
Introduction

ca.com



PV001

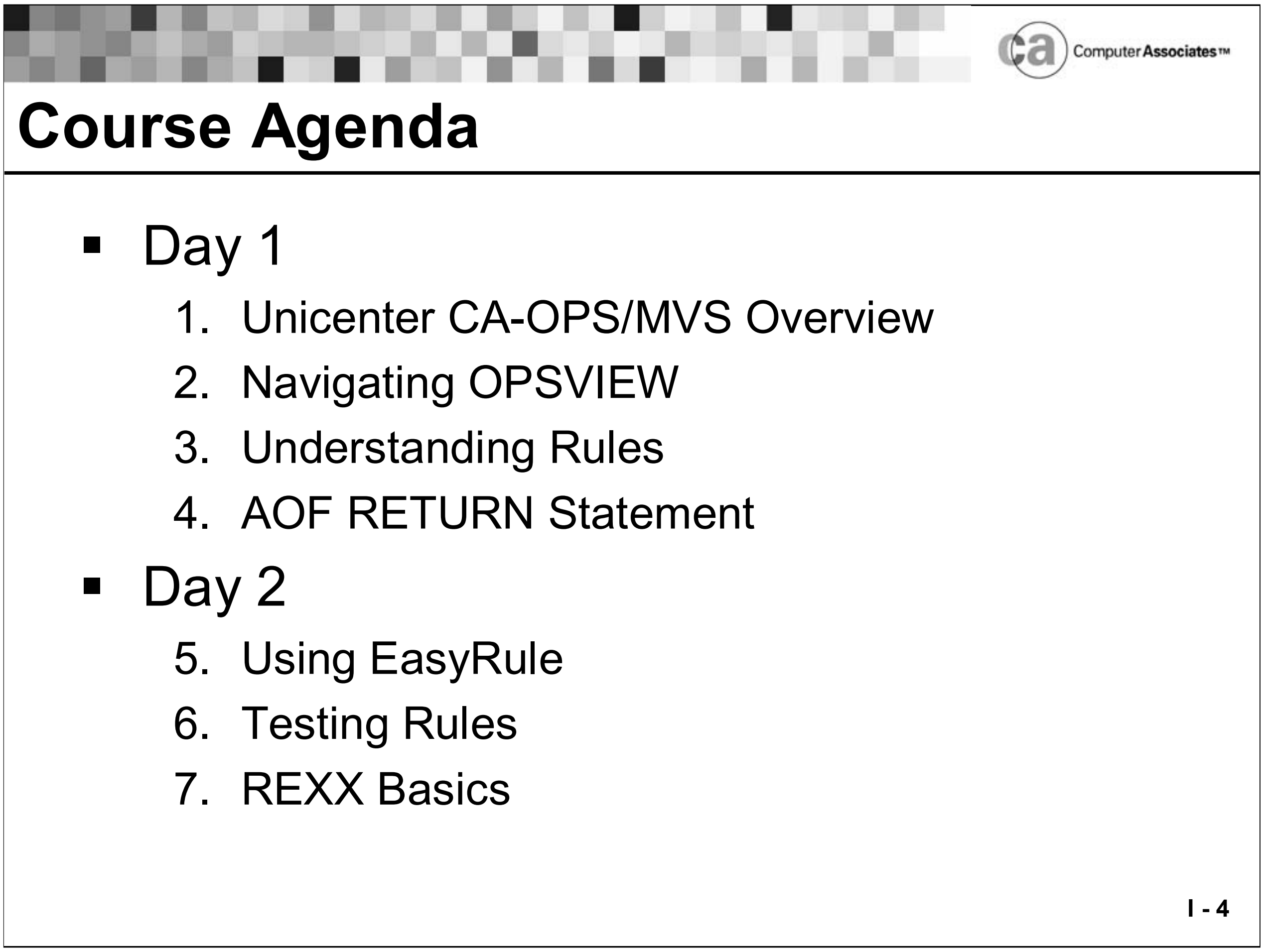
- Getting Started
  - 3 days
- Audience
  - Operations and systems personnel responsible for operating and maintaining Unicenter CA-OPS/MVS
  - Management personnel interested in learning more about Unicenter CA-OPS/MVS features and components
- Prerequisites
  - JCL, TSO, and ISPF familiarity
  - Computer operations experience



Course Objectives

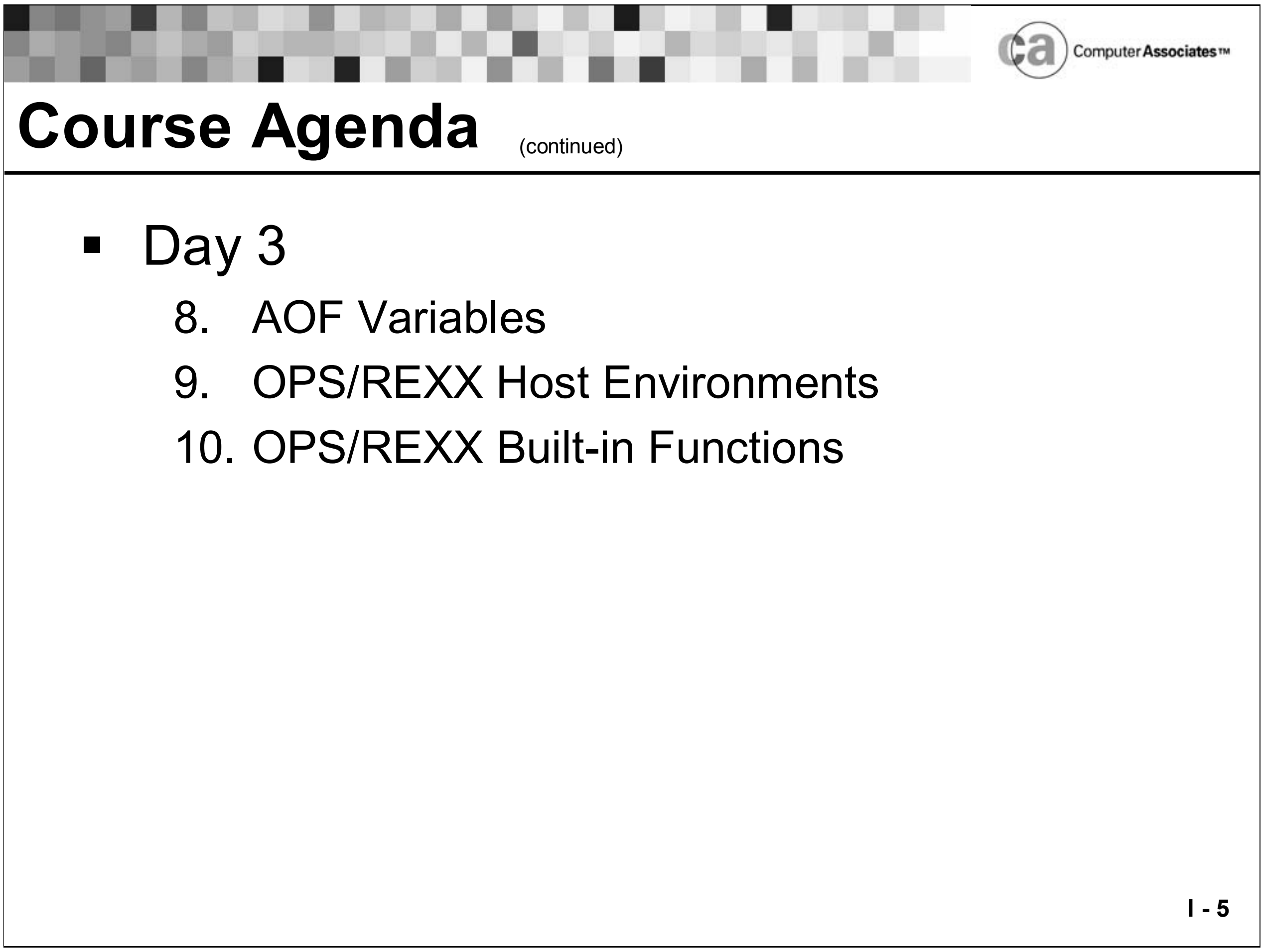
After this course, you will be able to:

- Describe Unicenter CA-OPS/MVS functions and components
- Navigate within OPSVIEW panels
- Write automation using EasyRule
- Test the automation you have written
- Describe the various tools that are available within Unicenter CA-OPS/MVS for creating specialized automation



Course Agenda

- Day 1
  1. Unicenter CA-OPS/MVS Overview
  2. Navigating OPSVIEW
  3. Understanding Rules
  4. AOF RETURN Statement
- Day 2
  5. Using EasyRule
  6. Testing Rules
  7. REXX Basics



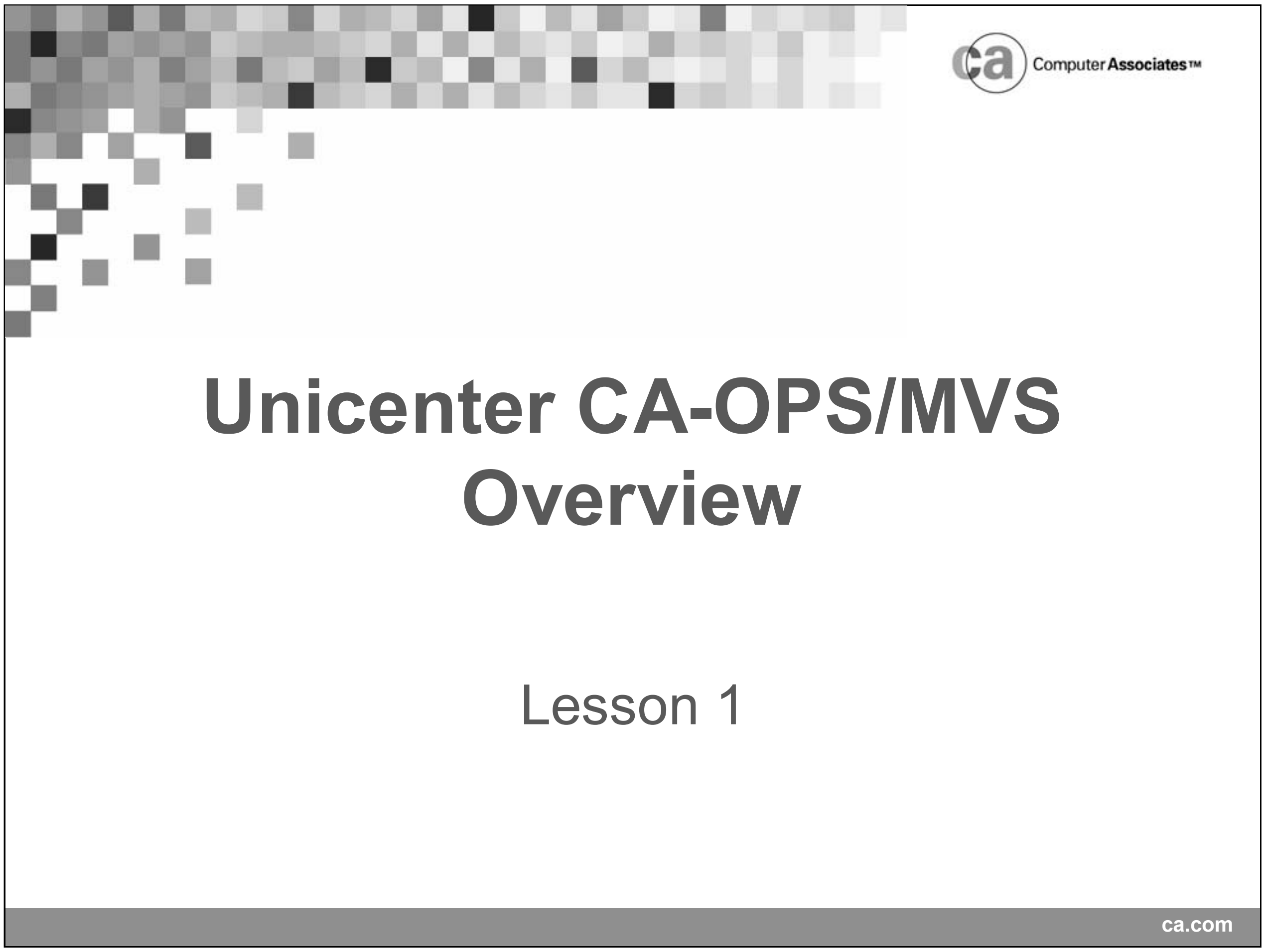
Course Agenda (continued)

- Day 3

  8. AOF Variables
  9. OPS/REXX Host Environments
  10. OPS/REXX Built-in Functions

I - 5

Notes:

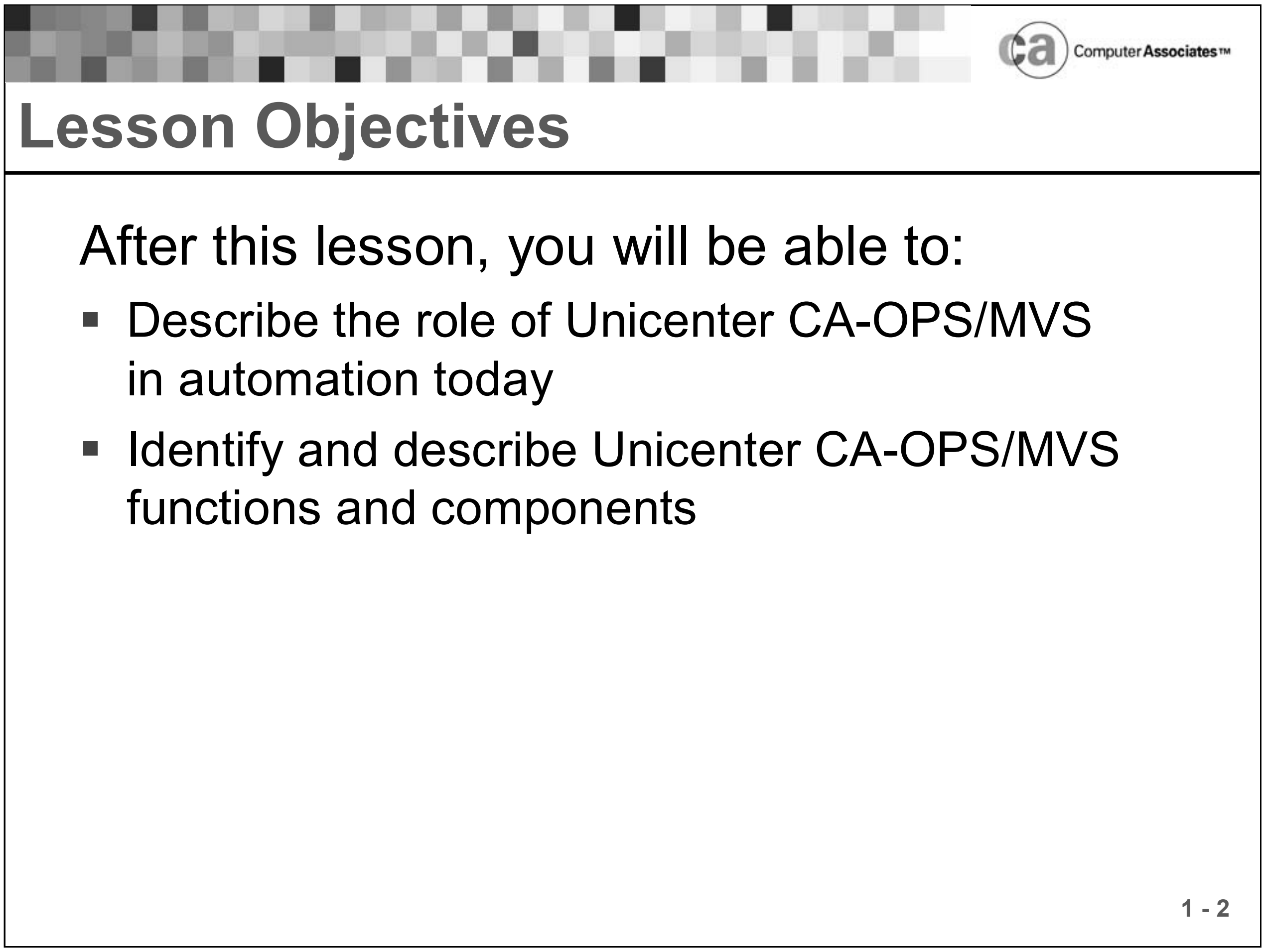


Computer Associates™

Unicenter CA-OPS/MVS Overview

Lesson 1

ca.com

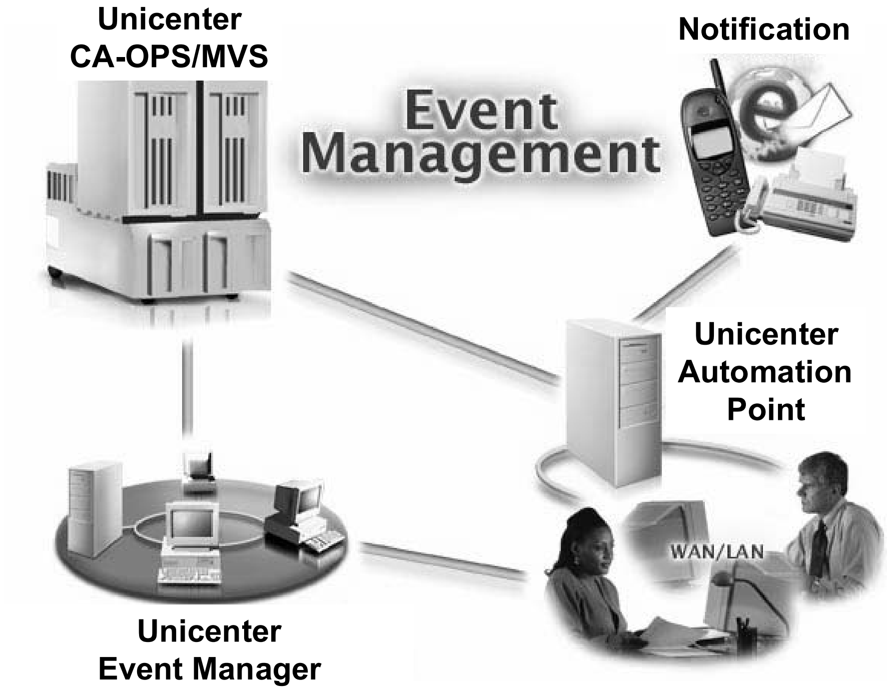


Lesson Objectives

After this lesson, you will be able to:

- Describe the role of Unicenter CA-OPS/MVS in automation today
- Identify and describe Unicenter CA-OPS/MVS functions and components

Automation Today



Unicenter CA-OPS/MVS

Notification

Unicenter Automation Point

Unicenter Event Manager and Agents

1 - 3

What Is Unicenter CA-OPS/MVS?

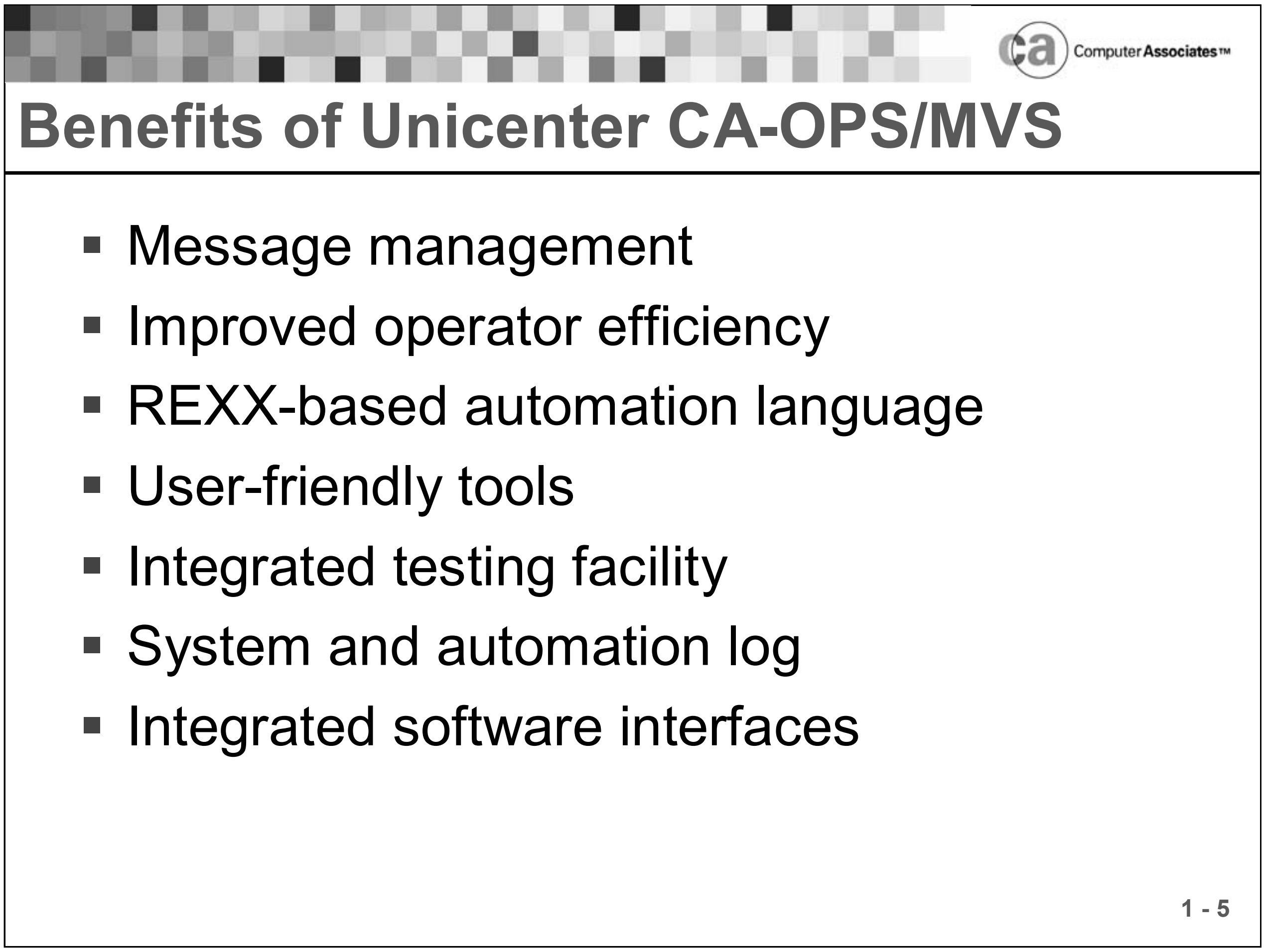
1 - 4

What Is Unicenter CA-OPS/MVS?

Unicenter CA-OPS/MVS is a comprehensive automated systems operations product for z/OS environments. It maximizes system availability, offers improved efficiency, reduces errors and downtime, and increases productivity on all levels.

Unicenter CA-OPS/MVS is the leading automation product in the mainframe market due to its dependable and efficient architecture, a powerful automation language, and various facilities that are geared at enabling rapid automation development. Its functionality provides infinite scalability to manage the smallest to the largest and most complex data centers without complex programming.

Unicenter CA-OPS/MVS plays a key role in CA's overall enterprise solution as a powerful standalone tool for ensuring maximum availability of the z/OS platform, as well as extending manageability and visibility in the enterprise.



Benefits of Unicenter CA-OPS/MVS

- Message management
- Improved operator efficiency
- REXX-based automation language
- User-friendly tools
- Integrated testing facility
- System and automation log
- Integrated software interfaces

1 - 5

Benefits of Unicenter CA-OPS/MVS

Message management leading to productivity gain:

- Highlight, make non-scrollable, route, reply to, and suppress messages.
- Treating messages as system events that should be responded to as installation policy dictates.
- Correlating multiple messages to produce a coherent picture of system status.

Improved operator efficiency by reducing human error:

- With a single command, an operator can trigger a complex series of commands that normally must be executed in a specific sequence.
- Controls and security checks on existing commands.

User-friendly facility for building automation:

- EasyRule is a powerful facility for building rules without programming. Operators can quickly create the rules needed for automation by simply filling in the blanks online.
- Automation specialists can view the OPS/REXX code that is generated by EasyRule. So while they build rules, they learn the REXX language.

Benefits of Unicenter CA-OPS/MVS (cont.)

User-friendly facility for building automation:

- EasyRule is a powerful facility for building rules without programming. Operators can quickly create the rules needed for automation by simply filling in the blanks online.
- Automation specialists can view the OPS/REXX code that is generated by EasyRule. So while they build rules, they learn the REXX language.

Integrated testing facility:

- Unicenter CA-OPS/MVS was designed to let you fully test automation before putting it into production. You can test a rule or subset of rules, and you can correct mistakes easily.

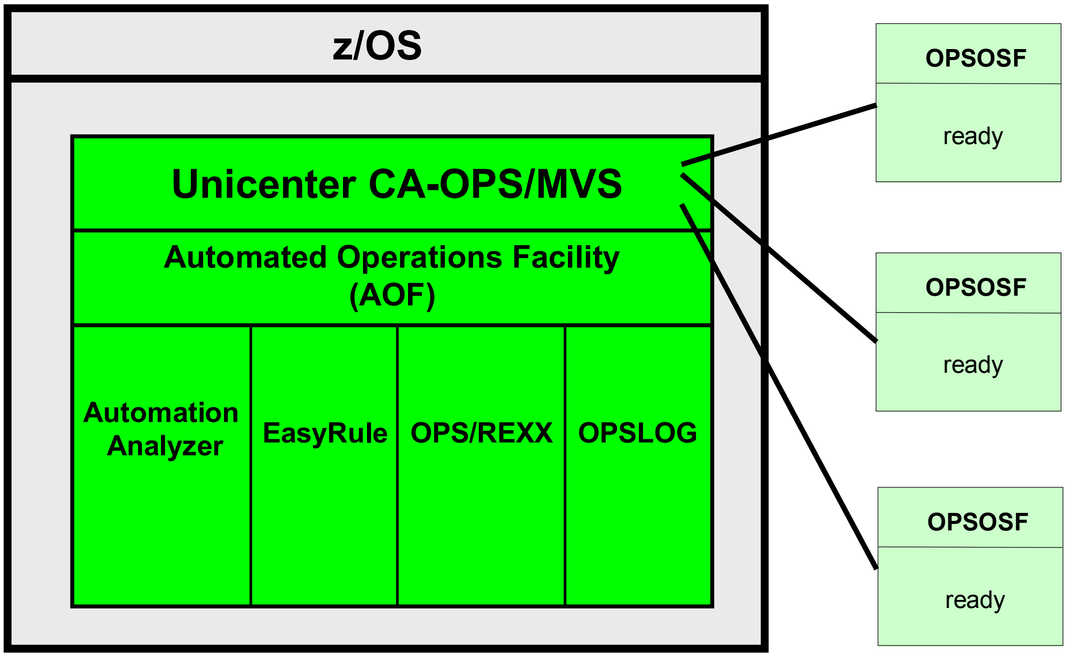
System and automation log:

- Many sites fear that activity will slip “under the covers” as they automate more. Unicenter CA-OPS/MVS guards against this by recording operational events into its system log file, OPSLOG, which you can quickly browse online. If they wish, users can interactively select messages associated with a particular job or time range.

Integrated software interfaces:

- External Product Interface (EPI) provides connectivity to all VTAM-based applications that users want to control via automation. You can use this interface, for instance, to request performance data from an online performance monitor or to log on to CICS to verify the availability of an application.

Unicenter CA-OPS/MVS Architecture



z/OS

Unicenter CA-OPS/MVS

Automated Operations Facility (AOF)

Automation Analyzer

EasyRule

OPS/REXX

OPSLOG

OPSOSF

ready

OPSOSF

ready

OPSOSF

ready

Unicenter CA-OPS/MVS Architecture

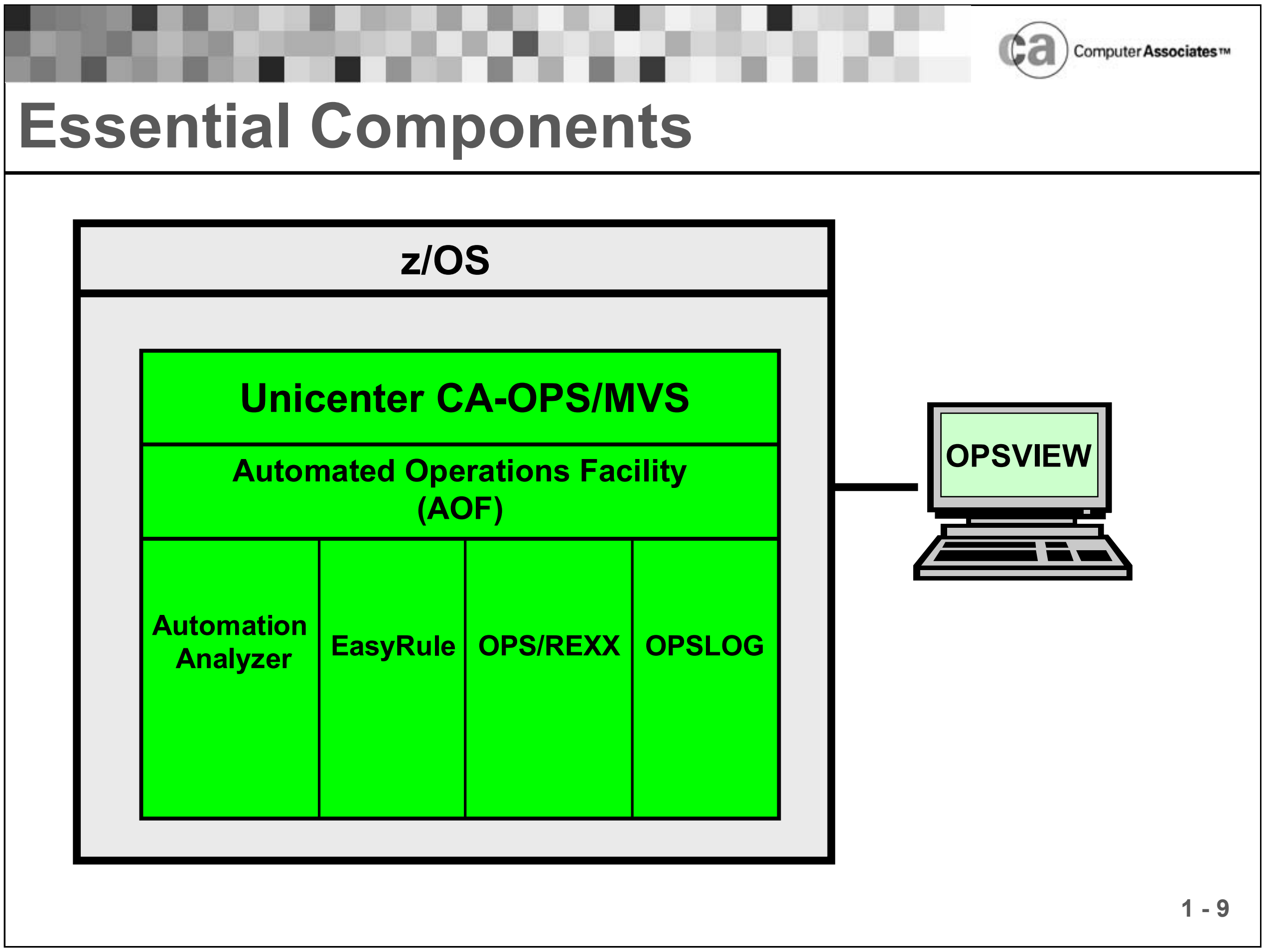
Unicenter CA-OPS/MVS has a multiple address space architecture. The central address space is called OPSMAIN. OPSMAIN is non-swappable. It is a formal z/OS subsystem. It puts a subsystem control table (SSCT) on the SSCT chain. This permits OPSMAIN to take advantage of the system exit points that z/OS provides.

Generally, OPSMAIN must be up for Unicenter CA-OPS/MVS facilities to be operative. OPSMAIN is typically started by an MVS START command in a member of the SYS1.PARMLIB data set. This means that the OPSMAIN address space can be brought up before JES, providing automation functionality early in the startup sequence.

One of Unicenter CA-OPS/MVS's most powerful features is its ability to process system events inline, or synchronously, via rules. This type of real-time automation is extremely effective and is possible because Unicenter CA-OPS/MVS rules execute in the address space in which an event occurs.

Unicenter CA-OPS/MVS Architecture (cont.)

The only types of automation you will not be able to perform within rules are those that require some interaction or any type of wait. These types of automation need to be performed by triggering an OPS/REXX program to a Unicenter CA- OPS/MVS server via the ADDRESS OSF host environment. The following diagram illustrates sending work to servers:



Essential Components

z/OS

Unicenter CA-OPS/MVS

Automated Operations Facility (AOF)

Automation Analyzer	EasyRule	OPS/REXX	OPSLOG

OPSVIEW

1 - 9

OPSVIEW

OPSVIEW is the operations interface for Unicenter CA-OPS/MVS. Typically, to interact with a computer system, you must type a long series of cryptic commands at a console. In response, the system returns terse messages that usually do not contain all the information that you need to manage the system. Such an environment may be difficult to understand and use and is often error-prone.

OPSVIEW combines the variety of Unicenter CA-OPS/MVS facilities, the OPS/REXX language, and the power of ISPF in an easy-to-use interface. OPSVIEW provides panels for performing various z/OS system functions, and it is the primary vehicle for controlling Unicenter CA-OPS/MVS itself. As its name suggests, OPSVIEW provides you with an operational view of your z/OS systems.

Automated Operations Facility (AOF)

The Automated Operations Facility monitors system events and responds to them automatically. With the AOF, you can program a response to a system event, such as a message or the passage of time. AOF rules are specially structured OPS/REXX programs that support automated operations by taking advantage of extensions made to the OPS/REXX language. The AOF is the “heart” of Unicenter CA-OPS/MVS.

Automation Analyzer

The Automation Analyzer shows where your data center can benefit most from automation. It studies message flow and then selects messages targeted for suppression and allows users to automatically create message suppression or response rules with a single keystroke.

EasyRule

EasyRule is a user-friendly, panel-driven facility that enables quick creation of automation rules using fill-in-the-blank ISPF panels. No programming or previous REXX knowledge is required. EasyRule generates SAA-compliant REXX code, enabling users to learn the REXX language while creating automation.

OPS/REXX

REXX is the standard common language for all of IBM’s environments under its Systems Application Architecture (SAA). CA chose REXX as the programming language for Unicenter CA-OPS/MVS because it is the most powerful and easiest to use command language available.

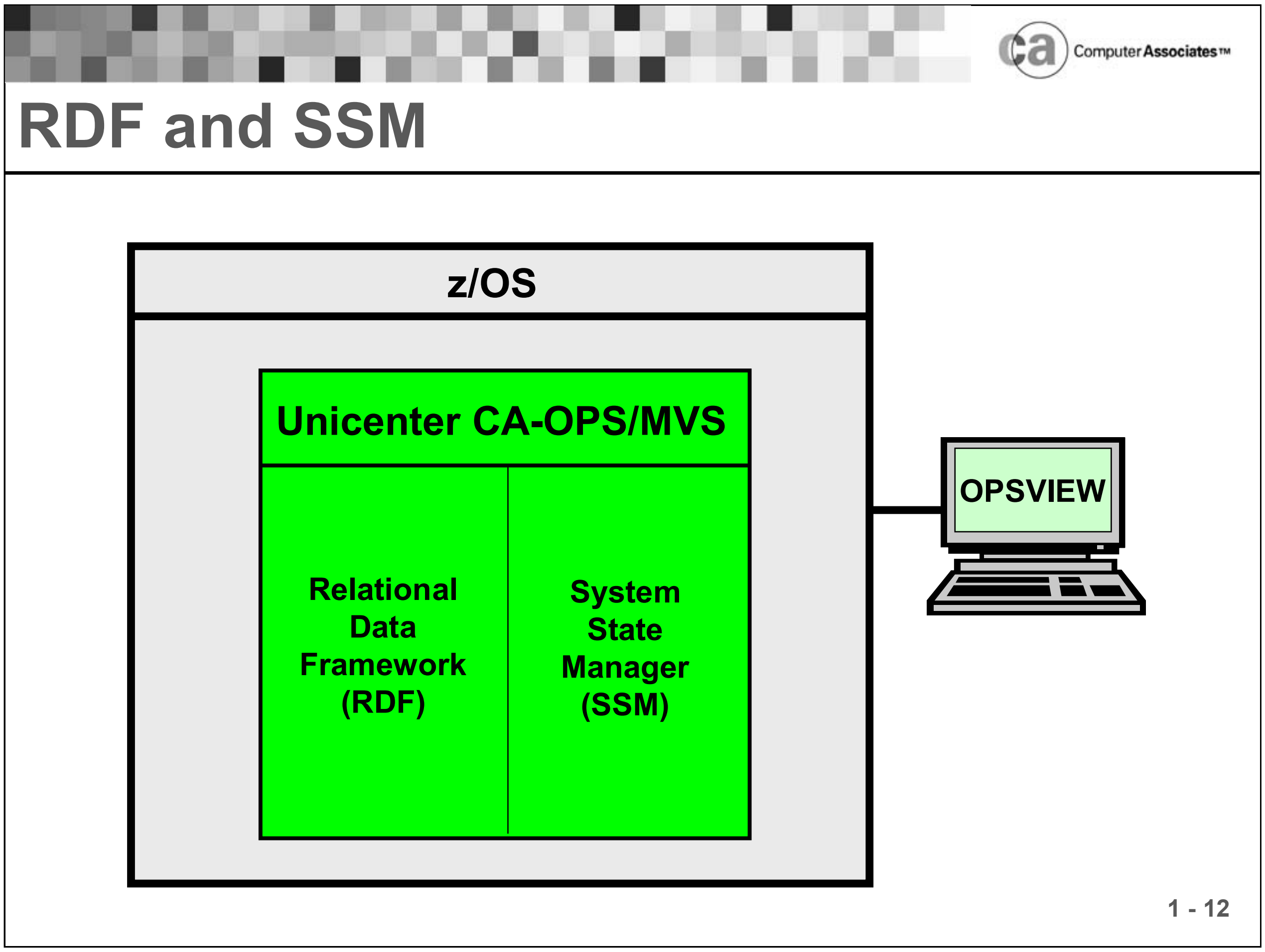
Unicenter CA-OPS/MVS comes with its own implementation of REXX called “OPS/REXX.” OPS/REXX provides a simple but capable high-level language to write operating system exits. It is a powerful, SAA-compliant programming language that adds to standard REXX a set of extensions that automate and enhance the productivity of z/OS operations. OPS/REXX is easy to understand and learn. If you can program in any language, you can learn to program in OPS/REXX.

OPSLOG

The OPSLOG is a repository that is used for analysis of all system events, including messages. OPSLOG has a browse facility that allows you to easily filter, search, archive, display, and print log data. Through OPSLOG’s customized view of the event log, users can quickly research and resolve problems. OPSLOG is a superior tool to SYSLOG.

OPSLOG

The OPSLOG is a repository that is used for analysis of all system events, including messages. OPSLOG has a browse facility that allows you to easily filter, search, archive, display, and print log data. Through OPSLOG's customized view of the event log, users can quickly research and resolve problems. OPSLOG is a superior tool to SYSLOG. Additionally, an OPSLOG WebView feature allows for OPSLOG access from a Windows platform. This OPSLOG WebView GUI design brings all the mainframe OPSLOG functionality to the workstation.



RDF and SSM

Relational Data Framework (RDF)

The Relational Data Framework facility lets you use Structured Query Language (SQL) statements to manage the large amounts of data required by automation rules and OPS/REXX programs. Instead of using large sets of variables, you can use the RDF to:

- Collect data.
- Organize data into a relational table that contains rows and columns of related information.
- Retrieve related system information by selecting it from a particular row or column.
- Update data in relational tables.

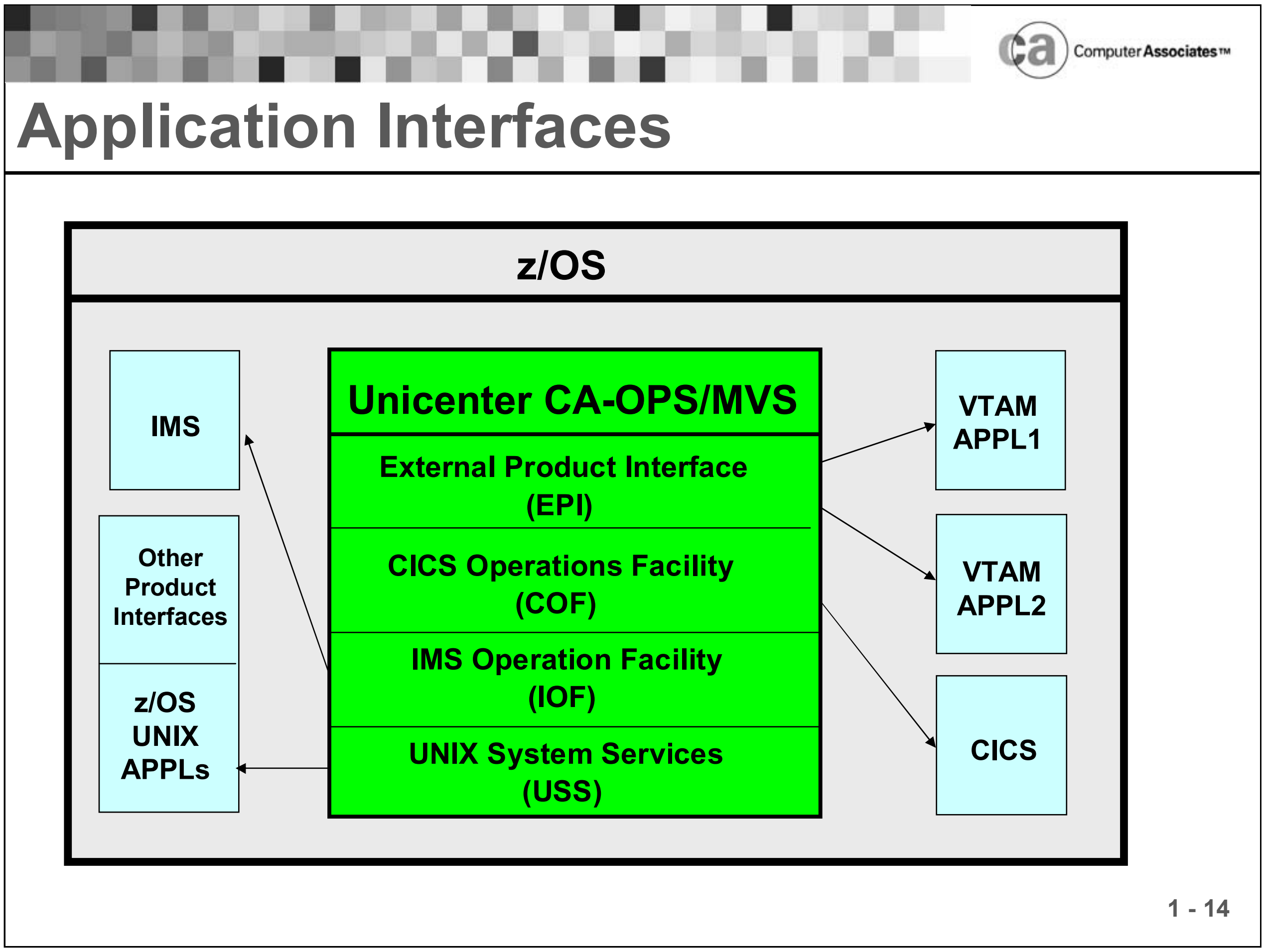
CA chose SQL to manage automation data because of SQL’s wide popularity with mainframe and PC users. The RDF consists of relational SQL tables plus a subset of the SQL language that conforms to American National Standards Institute (ANSI) standards. If you already know SQL, you should be able to use this subset of it right away.

System State Manager (SSM)

System State Manager monitors and controls the status of the hardware and software resources on your system. Using information from relational tables, System State Manager maintains a simple model of the proper state of your system resources. When the actual state of a resource deviates from that model (for instance, when a tape drive that should be online goes offline), System State Manager takes the necessary action to restore the resource to its proper state (for example, executes a synchronous OPS/REXX rule or issues a z/OS command), providing proactive and reactive state management of critical resources.

System State Manager's Snapshot function simplifies the process of setting up automation and maintaining started task resource data. The Snapshot function captures data about started tasks on the system and uses it to create a started task resource table. Using Snapshot greatly simplifies configuring and maintaining started task data.

System State Manager's Schedule Manager facility allows the desired state of resources to be defined by date, day of week, or time of day. The schedule override feature allows for the creation of temporary schedule overrides for managed resources.



Application Interfaces

z/OS
IMS
Other Product Interfaces
z/OS UNIX APPLs
Unicenter CA-OPS/MVS
External Product Interface (EPI)
CICS Operations Facility (COF)
IMS Operation Facility (IOF)
UNIX System Services (USS)
VTAM APPL1
VTAM APPL2
CICS

1 - 14

Application Interfaces

Unicenter CA-OPS/MVS has numerous interfaces that allow you to perform various functions. For a comprehensive list, refer to your product documentation. The following interfaces are discussed in this lesson:

- External Product Interface (EPI)
- CICS Operations Facility (COF)
- IMS Operation Facility (IOF)
- UNIX System Services (USS)

Note: The COF and IOF are separately licensed optional facilities.

External Product Interface (EPI)

The External Product Interface permits Unicenter CA-OPS/MVS systems that are running under VTAM to communicate with any VTAM application that supports IBM 3270-type virtual terminals. The EPI appears to VTAM as a real 3270 terminal that can emulate any number of 3270-type virtual terminals that are connected to any number of VTAM applications.

CICS Operations Facility (COF)

The CICS Operations Facility is an optional interface to CICS that extends the capability for AOF rule processing to CICS messages, which are written only to CICS transient data queues. This additional message traffic expands the number of events that you can use to control CICS subsystems. Terminal failures, users' logon and logoff activities, and journal switches are just a few of the CICS events that will become visible to AOF rules that are using the COF. With the COF interface installed, a single copy of Unicenter CA-OPS/MVS can handle an unlimited number of CICS regions.

IMS Operation Facility (IOF)

The IMS Operation Facility is an optional interface to IMS that extends automation facilities to IMS. For example, you can write AOF rules that process IMS messages, and you can use OPSVIEW to operate IMS. A single copy of Unicenter CA-OPS/MVS can handle up to 32 copies of IMS at any combination of IBM-supported IMS levels.

UNIX System Services (USS)

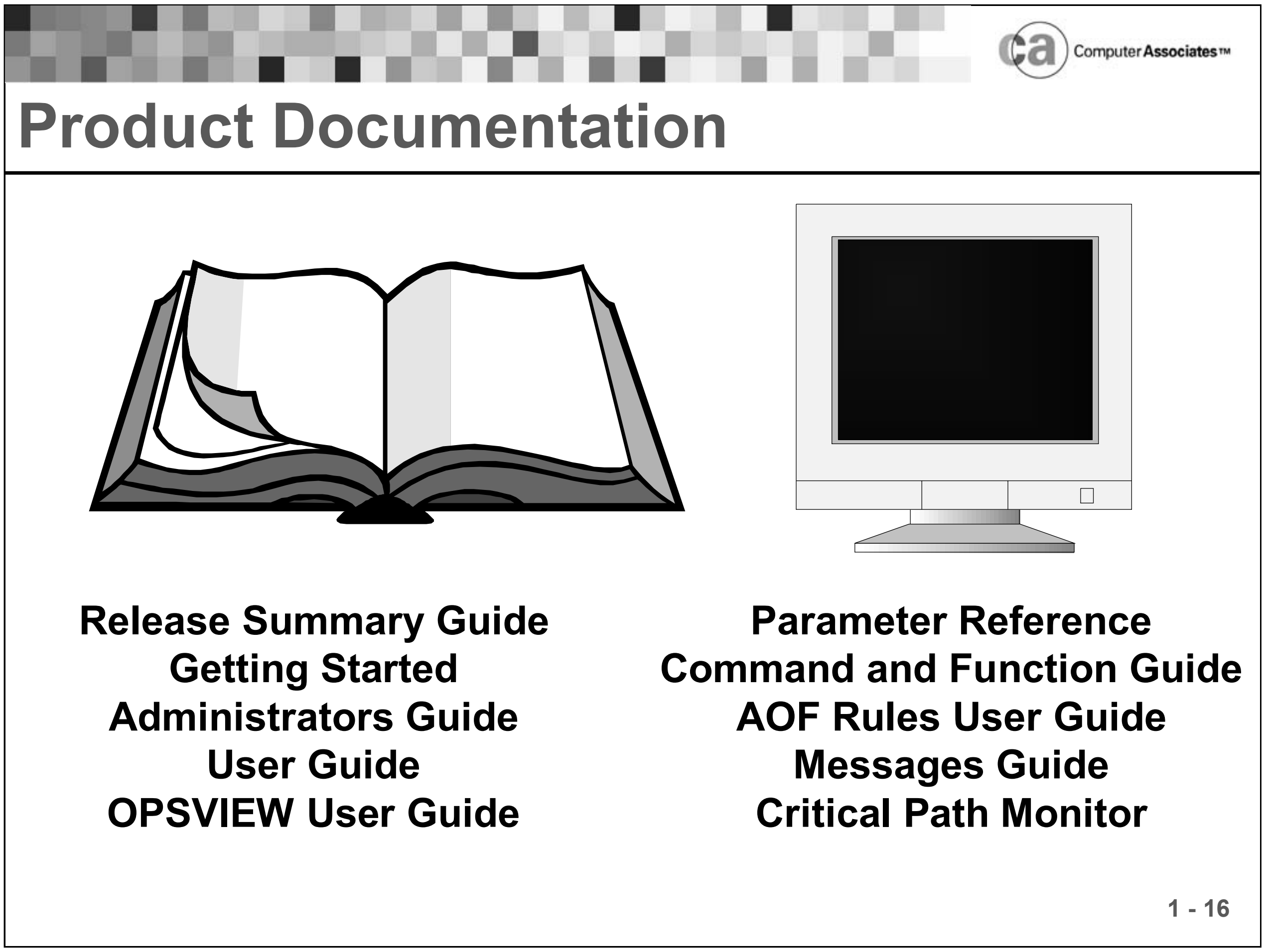
The UNIX System Services provides comprehensive, bi-directional integration with Unicenter CA-OPS/MVS. This provides for the management of the USS environment on the z/OS platform. An additional USS process event feature provides system automation events for the initialization and termination of USS processes.

Other Product Interfaces

Unicenter CA-OPS/MVS also includes comprehensive interfaces to other z/OS software that you may be using, including:

- Unicenter CA-7
- Unicenter CA-Jobtrac
- Unicenter CA-Scheduler
- Unicenter CA-Netmaster
- Unicenter CA-SYSVIEW
- NetView
- MVS/QuickRef
- OMEGAMON

For details about these interfaces, refer to your product documentation.



Product Documentation

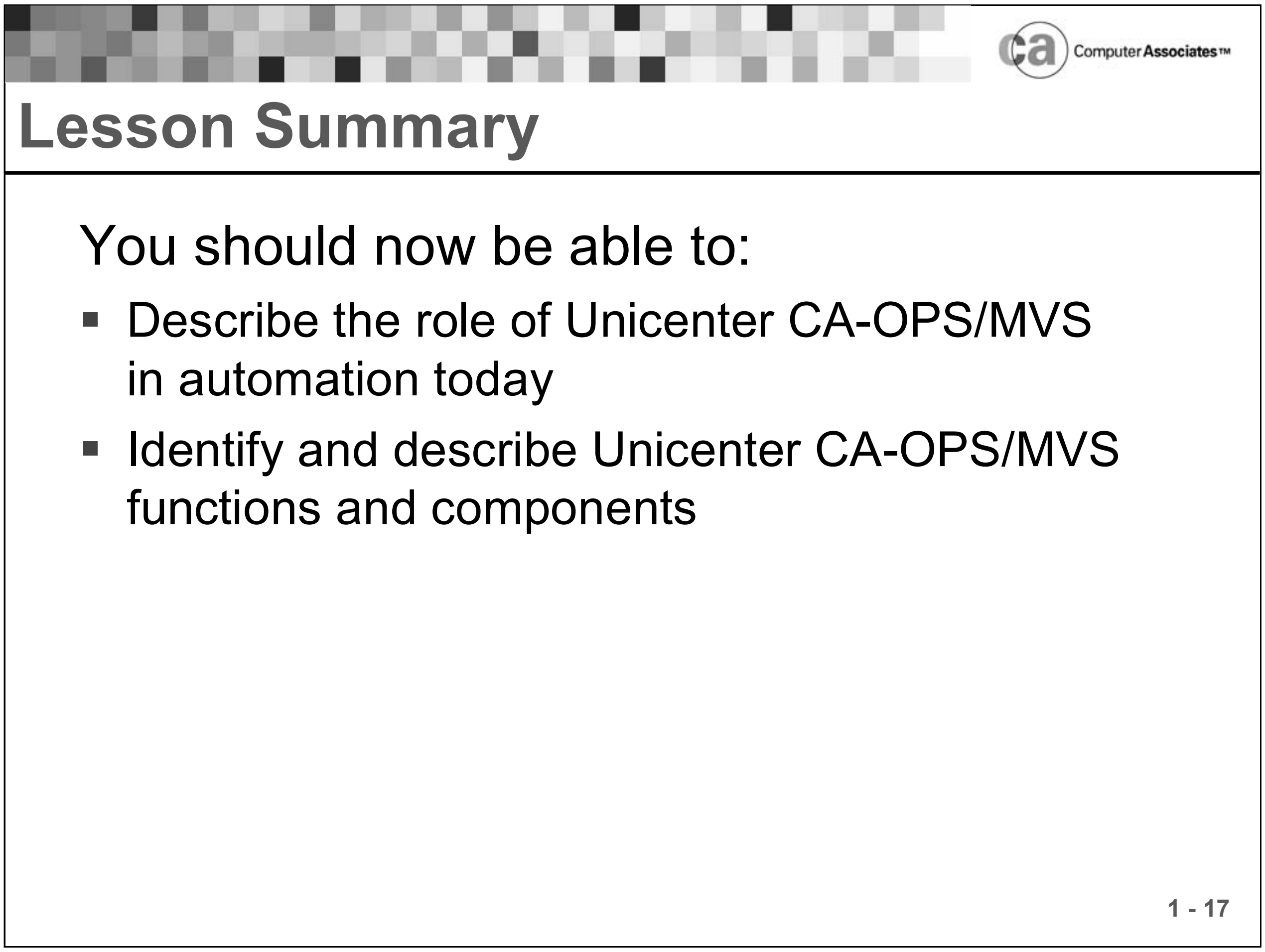
Release Summary Guide
Getting Started
Administrators Guide
User Guide
OPSVIEW User Guide

Parameter Reference
Command and Function Guide
AOF Rules User Guide
Messages Guide
Critical Path Monitor

1 - 16

Product Documentation

The Unicenter CA-OPS/MVS documentation set is provided in Portable Document Format (PDF) and BookManager online formats. The Getting Started guide is also provided in hard copy form.



Lesson Summary

You should now be able to:

- Describe the role of Unicenter CA-OPS/MVS in automation today
- Identify and describe Unicenter CA-OPS/MVS functions and components

Lesson 1 Assessment

1 - 18

Lesson 1 Assessment

Match the descriptions below with one of the following terms:

OPSVIEW
SYSLOG
COF
POI
RDF
CICS
OPSLOG
Automation Point
AOF
EasyRule
OPS/REXX
REXX
Unicenter CA-OPS/MVS
Unicenter
CA-SYSVIEW/E
EPI

__________ Unicenter CA-OPS/MVS repository of system events.

__________ A facility for creating rules.

__________ CA's distributed automation offering.

__________ SAA-compliant programming language used in Unicenter CA-OPS/MVS.

__________ Operations interface for Unicenter CA-OPS/MVS.

__________ Automated systems operations product for z/OS environments.

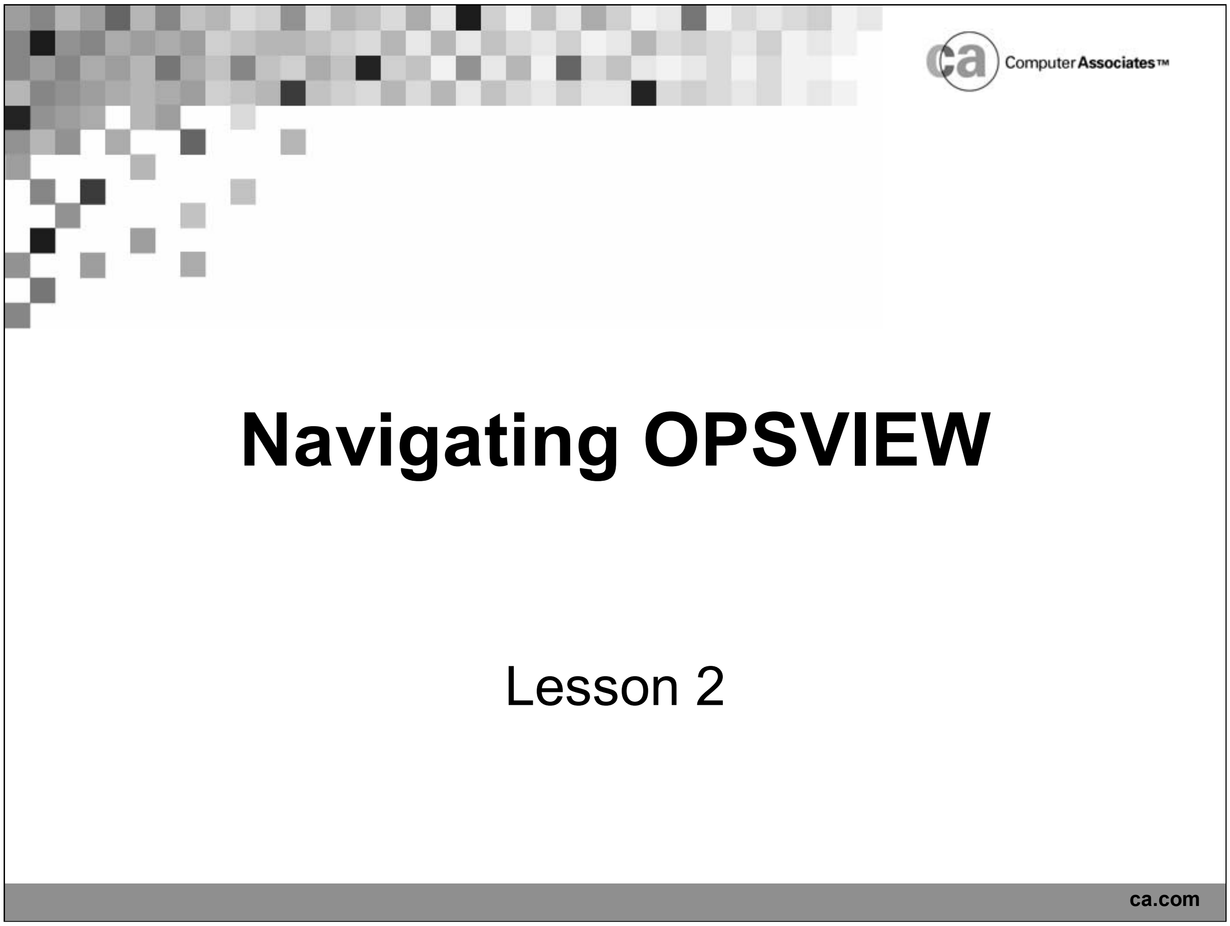
__________ Enables communications with VTAM applications.

__________ Unicenter CA-OPS/MVS interface to CICS.

__________ A facility that lets you use SQL statements to manage data.

__________ The heart of Unicenter CA-OPS/MVS.

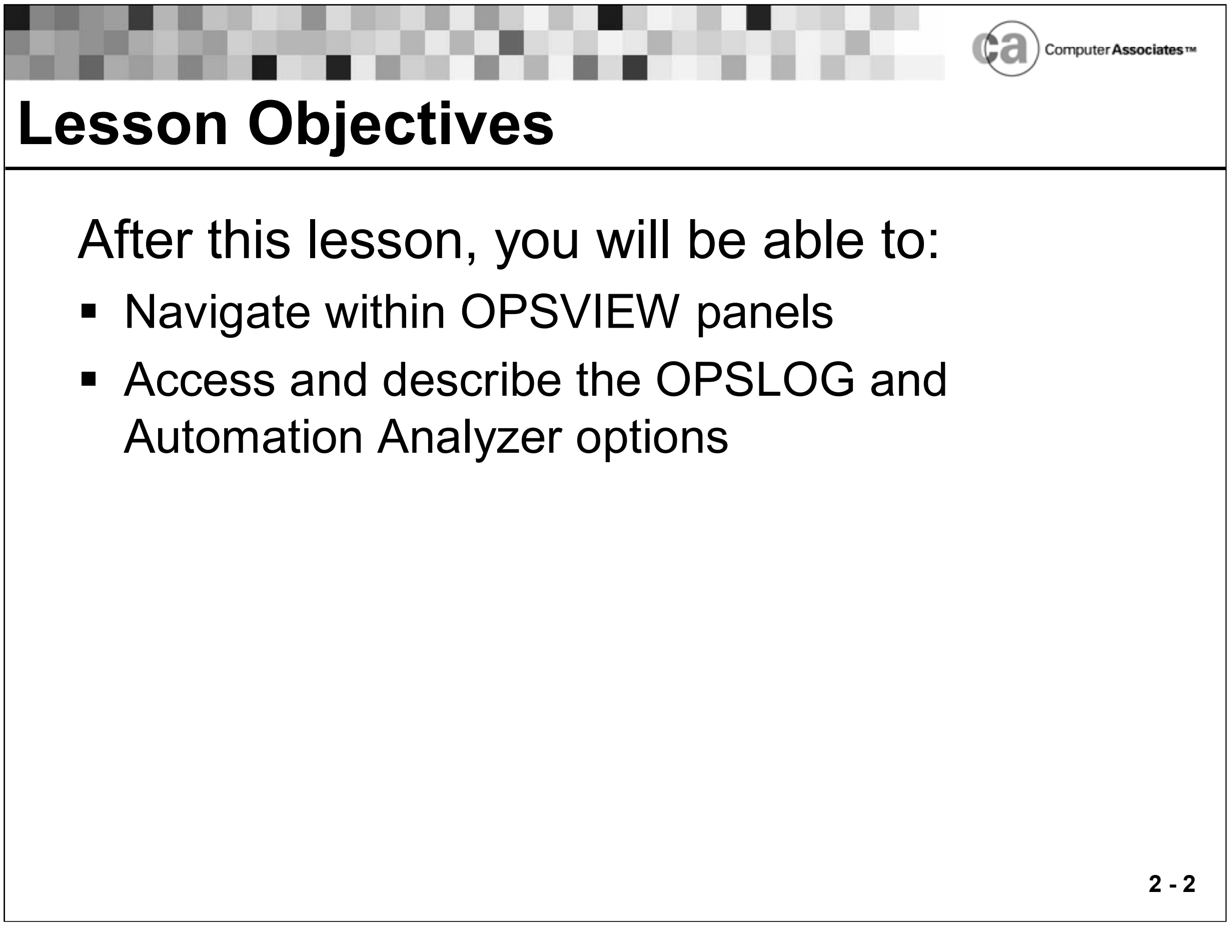
Notes:



Navigating OPSVIEW

Lesson 2

ca.com

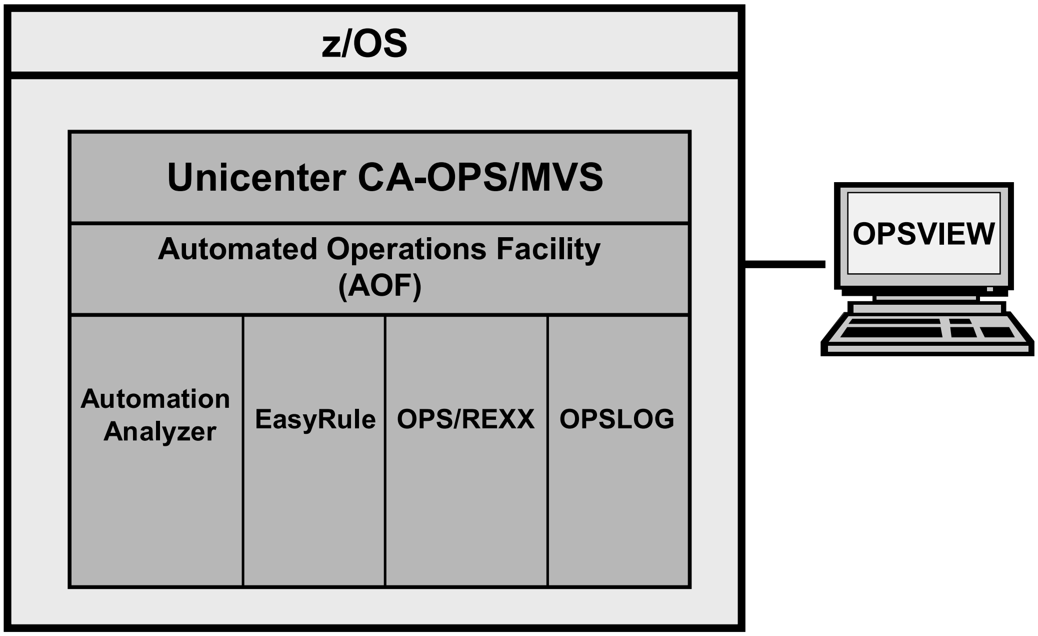


Lesson Objectives

After this lesson, you will be able to:

- Navigate within OPSVIEW panels
- Access and describe the OPSLOG and Automation Analyzer options

OPSVIEW Basics



2 - 3

OPSVIEW Basics

OPSVIEW is the operations interface for Unicenter CA-OPS/MVS. It provides panels for performing various z/OS system functions, and it is the primary vehicle for controlling Unicenter CA-OPS/MVS itself.

There are two ways in which you can use OPSVIEW:

- Use the facilities OPSVIEW provides.

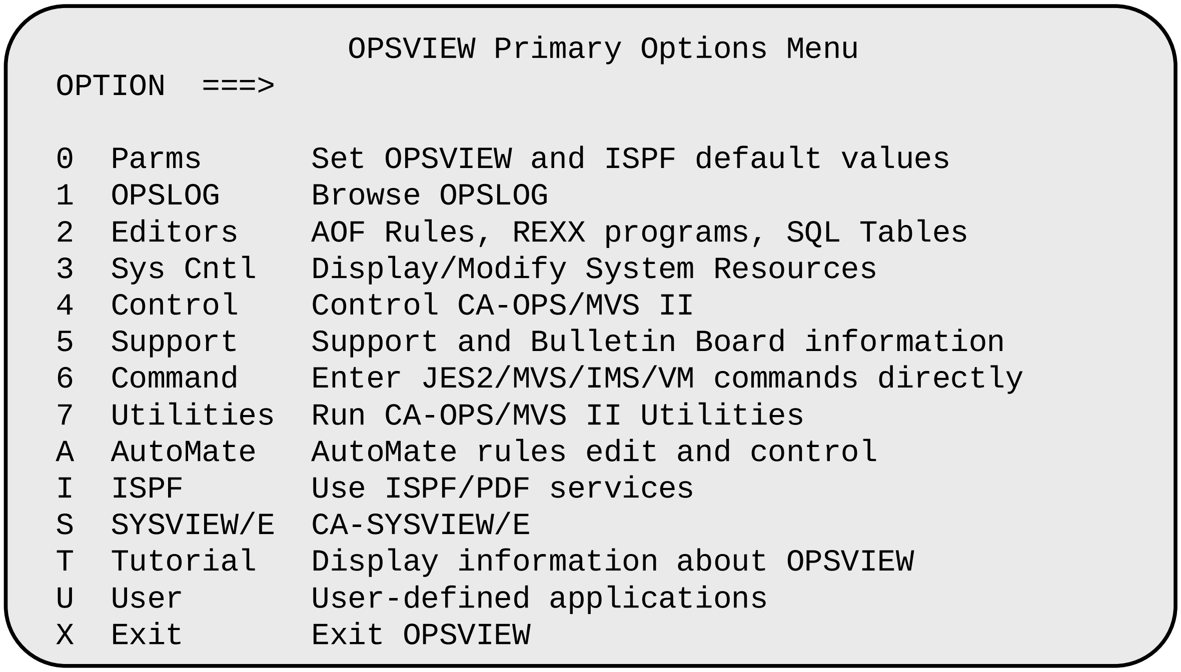
  Since TSO's interactive facilities are available with OPSVIEW, you can accomplish any operational procedure using OPSVIEW that you can using a z/OS console. Using OPSVIEW makes these procedures easier to perform.

- Write your own OPSVIEW applications.

  Because OPSVIEW is written almost entirely in the TSO CLIST and OPS/REXX languages, you can easily extend it with your own programs.

  You can customize or modify the sample programs that are provided with Unicenter CA-OPS/MVS to meet your site's unique requirements. See your product documentation for descriptions of these programs.

OPSVIEW Primary Options Menu

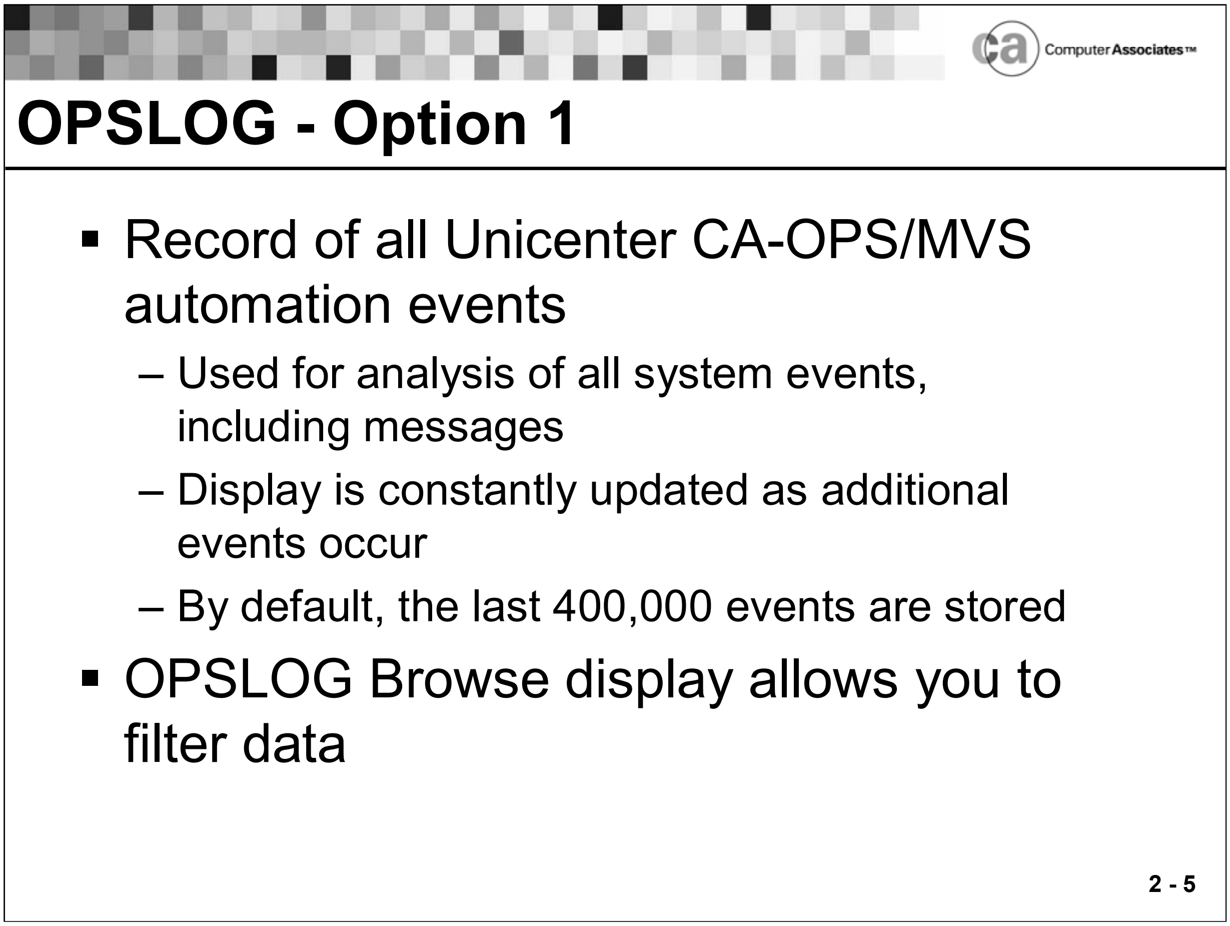


```
                OPSVIEW Primary Options Menu
OPTION  ===>

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

OPSVIEW Primary Options Menu

The first menu you see is the OPSVIEW Primary Options Menu. The Primary Options Menu contains a list of other menus, from which you can perform various functions.

To select an option, enter its option code in the Option field. For example, to browse OPSLOG, type 1 in the Option field and then press Enter.



OPSLOG - Option 1

- Record of all Unicenter CA-OPS/MVS automation events
  - Used for analysis of all system events, including messages
  - Display is constantly updated as additional events occur
  - By default, the last 400,000 events are stored
- OPSLOG Browse display allows you to filter data

2 - 5

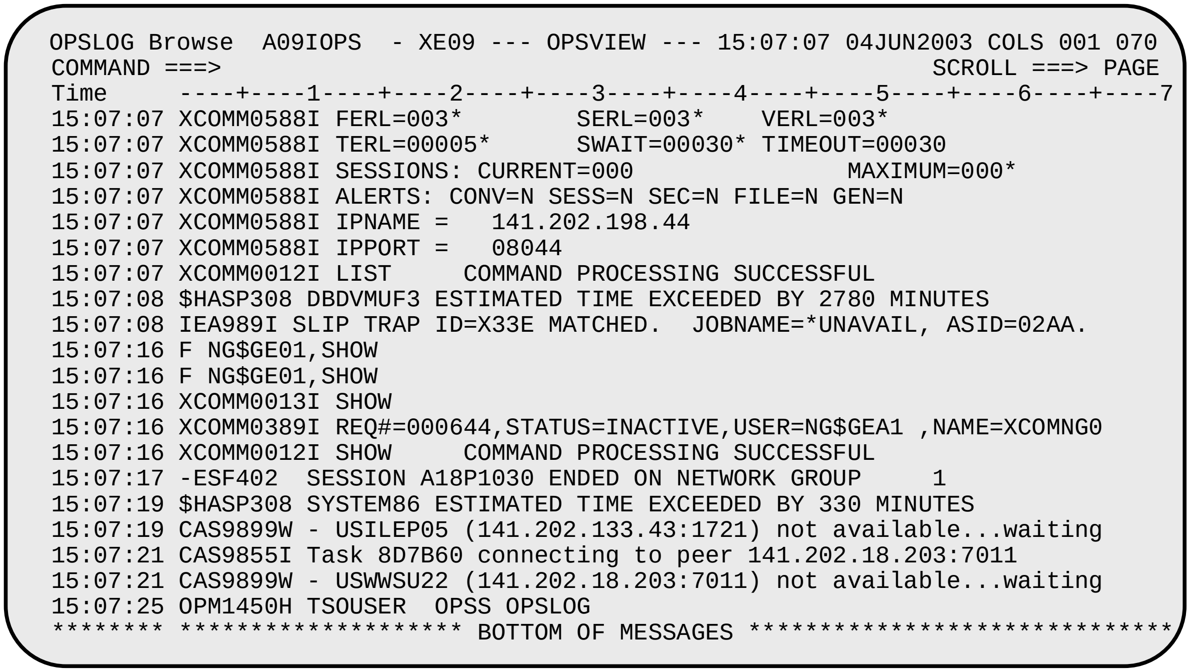
OPSLOG

The OPSLOG is a record of all automation events. As you learned in the last lesson, OPSLOG is a repository that is used for analysis of all system events, including messages.

In its OPSLOG, Unicenter CA-OPS/MVS keeps copies of all automation events. The display is constantly updated as additional events occur. OPSLOG has a browse facility that allows you to easily filter, search, archive, display, and print log data.

The OPSLOG resides in the extended private area of the Unicenter CA-OPS/MVS main product address space (OPSMAIN). You control the number of events that are kept in OPSLOG via the BROWSEMAX parameter. By default, Unicenter CA-OPS/MVS stores the last 400,000 messages. In practice, the OPSLOG may contain as few as 10,000 messages or as many as 3,000,000 messages.

OPSLOG Browse



```
OPSLOG Browse  A09IOPS  - XE09 --- OPSVIEW --- 15:07:07 04JUN2003 COLS 001 070
COMMAND ===>                                                  SCROLL ===> PAGE
Time     ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
15:07:07 XCOMM0588I FERL=003*        SERL=003*    VERL=003*
15:07:07 XCOMM0588I TERL=00005*      SWAIT=00030* TIMEOUT=00030
15:07:07 XCOMM0588I SESSIONS: CURRENT=000               MAXIMUM=000*
15:07:07 XCOMM0588I ALERTS: CONV=N SESS=N SEC=N FILE=N GEN=N
15:07:07 XCOMM0588I IPNAME =   141.202.198.44
15:07:07 XCOMM0588I IPPORT =   08044
15:07:07 XCOMM0012I LIST     COMMAND PROCESSING SUCCESSFUL
15:07:08 $HASP308 DBDVMUF3 ESTIMATED TIME EXCEEDED BY 2780 MINUTES
15:07:08 IEA989I SLIP TRAP ID=X33E MATCHED.  JOBNAME=*UNAVAIL, ASID=02AA.
15:07:16 F NG$GE01,SHOW
15:07:16 F NG$GE01,SHOW
15:07:16 XCOMM0013I SHOW
15:07:16 XCOMM0389I REQ#=000644,STATUS=INACTIVE,USER=NG$GEA1 ,NAME=XCOMNG0
15:07:16 XCOMM0012I SHOW     COMMAND PROCESSING SUCCESSFUL
15:07:17 -ESF402  SESSION A18P1030 ENDED ON NETWORK GROUP     1
15:07:19 $HASP308 SYSTEM86 ESTIMATED TIME EXCEEDED BY 330 MINUTES
15:07:19 CAS9899W - USILEP05 (141.202.133.43:1721) not available...waiting
15:07:21 CAS9855I Task 8D7B60 connecting to peer 141.202.18.203:7011
15:07:21 CAS9899W - USWWSU22 (141.202.18.203:7011) not available...waiting
15:07:25 OPM1450H TSOUSER  OPSS OPSLOG
******** ******************** BOTTOM OF MESSAGES *******************************
```

Accessing OPSLOG

You can access OPSLOG from the OPSVIEW Primary Options Menu. Additionally, the OPSLOG can be accessed via the OPSLOG Webview. The OPSLOG Webview consists of two components: a browser (GUI) and a server. The design of the GUI brings all the OPSLOG functionality available via OPSVIEW option 1 to the workstation. The OPSLOG Webview is a base component of the product, but involves additional installation for proper implementation.

This class will utilize Option 1 from the OPSVIEW Primary options menu to access and demonstrate the use of the OPSLOG.

The above display is an example of a typical OPSLOG Browse panel. Notice the information shown on the top line of the panel. It can help you determine the status of OPSLOG Browse. On the example panel:

- “A09IOPS” is the MSF ID. This field indicates the MSF ID of the system whose OPSLOG you are browsing. The value in this field helps you discern whether you are viewing the local system’s OPSLOG or a remote system’s OPSLOG.
- “XE09” is the SMF ID. This field indicates the SMF ID of the operating system to which you are logged on.
- “15:07:07 04JUN2003” indicates the date and time of the first automation event, in the form hh:mm:ss ddmmmyyyy.

Moving Around Within OPSLOG

To move around in the OPSLOG Browse display, use the standard ISPF Up, Down, Left, and Right PF keys. Use the Scroll field to change the scroll amount.

Follow these guidelines when navigating through the OPSLOG event stream:

- When you first enter OPSLOG Browse, the display is positioned at the bottom of the OPSLOG event stream. Press Enter to refresh the display with the latest events. The “Bottom of Messages” marker is visible.
- If the OPSLOG Browse display is positioned at the top of the event stream and the data area is full, press Enter to cause the oldest events to disappear from the display. This occurs to accommodate the addition of new events to the data area. The “Top of Messages” marker is visible at the top of the display.
- If you move the display from its initial position at the bottom of the event stream, it will not return to the bottom of the even stream when you press Enter. To put the display back into the mode in which it moves to the bottom of the event stream each time you press Enter, type M on the command line, and press the Down PF key.

Displaying Extra Columns of Information

You can use the DISPLAY command to change the format of the OPSLOG Browse display. By issuing this command, you tell Unicenter CA-OPS/MVS what extra columns of information you want to view for each event that appears on the OPSLOG Browse display. The settings of the OPSLOG Browse display columns are retained across OPSLOG Browse sessions. See your documentation for details about this command.

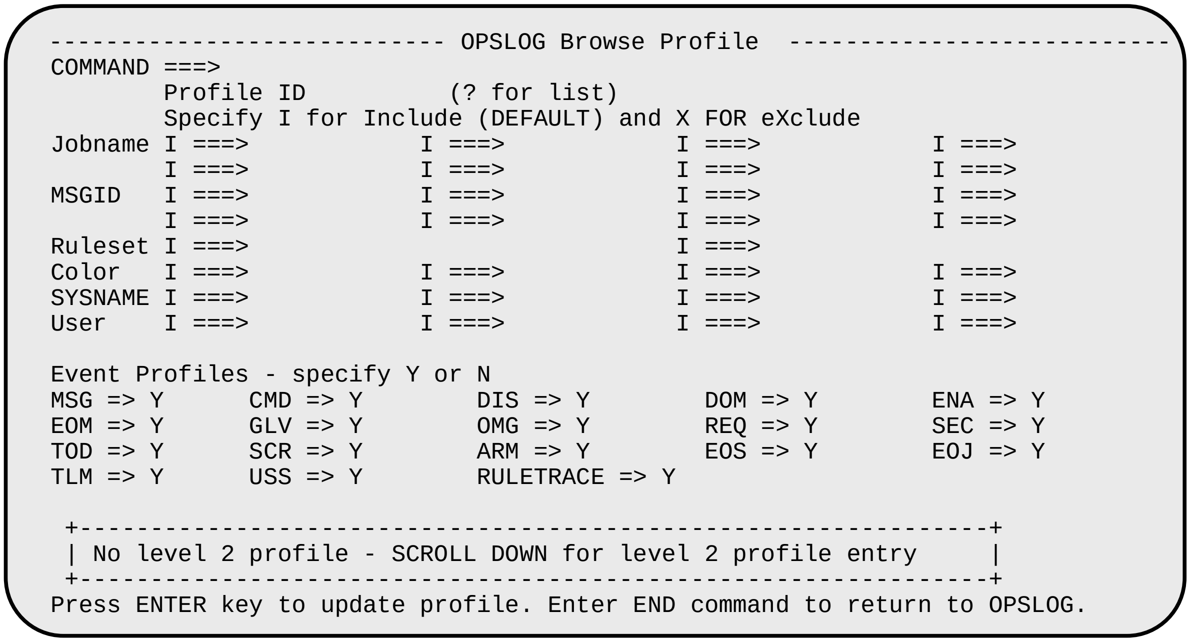
Finding Character Strings in OPSLOG Browse

You can use the OPSLOG Browse FIND command to locate character strings within event text.

You can use the OPSLOG Browse FIND column command to locate character strings within a specific OPSLOG Browse display column.

The OPSLOG Browse FIND command works very much like the ISPF FIND command. See your OPS/MVS documentation for more information.

OPSLOG Browse Profile



```
---------------------------- OPSLOG Browse Profile  --------------------------
COMMAND ===>
        Profile ID          (? for list)
        Specify I for Include (DEFAULT) and X FOR eXclude
Jobname I ===>            I ===>            I ===>            I ===>
        I ===>            I ===>            I ===>            I ===>
MSGID   I ===>            I ===>            I ===>            I ===>
        I ===>            I ===>            I ===>            I ===>
Ruleset I ===>                              I ===>
Color   I ===>            I ===>            I ===>            I ===>
SYSNAME I ===>            I ===>            I ===>            I ===>
User    I ===>            I ===>            I ===>            I ===>

Event Profiles - specify Y or N
MSG => Y      CMD => Y        DIS => Y        DOM => Y        ENA => Y
EOM => Y      GLV => Y        OMG => Y        REQ => Y        SEC => Y
TOD => Y      SCR => Y        ARM => Y        EOS => Y        EOJ => Y
TLM => Y      USS => Y        RULETRACE => Y

 +-----------------------------------------------------------------+
 | No level 2 profile - SCROLL DOWN for level 2 profile entry     |
 +-----------------------------------------------------------------+
Press ENTER key to update profile. Enter END command to return to OPSLOG.
```

2 - 8

OPSLOG Browse Profile

Because the amount of automation events that you can view using OPSLOG Browse can be very large, you can set an OPSLOG Browse profile to filter out some of them. By doing so, you can browse a subset of events that includes only the type of events you want to see.

Specifying Profile Criteria

There are several places you can specify your profile criteria:

- In the fields on the OPSLOG Browse Profile panel (shown above)

  You may find it easier to set your profile from the OPSLOG Browse Profile panel if you are a new user of Unicenter CA-OPS/MVS or you are unfamiliar with OPSLOG Profile options. The OPSLOG Browse Profile panel provides fields for setting each of the options, which means you will not have to depend upon your memory to know which options are available.

- On the command line on the primary OPSLOG Browse panel

  You may prefer to set your profile criteria directly on the primary OPSLOG Browse panel if you are familiar with the options.

- From OPSVIEW option 0.4

  OPSVIEW option 0.4 allows you to change or clear the profile before you enter OPSLOG Browse.
- To access the OPSLOG Browse profile panel, issue the PROFILE command from the command line on the primary OPSLOG Browse panel. You see a display similar to the one shown on the slide.

The options you choose on the OPSLOG Browse Profile panel determine which automation events OPSLOG presents. For example:

- If you enter a job name in the Jobname field with an “I” (Inclusion), only events with that job name appear on the OPSLOG Browse display. If you enter an “X” before the job name, the opposite occurs; all other events appear in the OPSLOG Browse display, with the exception of those events that have the entered job name.
- All e(X)clusion entries are processed first. Only then are any (I)nclusion entries applied. For example, if the job name JOB771 is entered with an “I,” and an MSGID of $HASP373 is entered with an “X,” no events with a message ID of $HASP373 appear in OPSLOG. Only events with the job name of JOB771 (except for $HASP373) appear.
- For some individual options (such as Ruleset, Color, and User), you may specify multiple values. These multiple values are linked with a logical “or”; thus, if you specify two values for Ruleset, the events that appear in OPSLOG Browse are those that were initiated by either rule set.
- All profile options are linked to the other options by a logical “and.” Thus, if you specify a value for the Jobname field and a value for the Msgid field, only those events that fit both criteria appear on the OPSLOG Browse display. However, if you specify two Jobname values (such as VTAM and TEST) and one Msgid value (such as IEF250I), those events that fit either set of criteria (a job name of VTAM and a message ID of IEF250I, or a job name of TEST and a message ID of IEF250I) appear on the OPSLOG Browse display.
- If you specify values that do not match any OPSLOG entries (for example, a job name that never existed in the system), it can affect system performance because the entire OPSLOG data area must be searched. If this happens, OPS/MVS must reference many pages of virtual storage, which causes many real storage pages to be assigned.

Setting Options for the OPSLOG Browse Profile Panel

There are three basic types of options on the OPSLOG Browse Profile panel:

- Nonevent-related
- Event-related
- Command-only options

Nonevent-related OPSLOG Browse Profile Options:

- Jobname – Limits events to those that are produced by this job. You can specify up to eight job names.
- MSGID – Limits events to those that have this message ID. You can specify up to eight message IDs.
- Ruleset – Limits events to those that are processed by this ruleset.rule. You can specify up to two rule sets.
- Color – Limits events to those that display in this color. You can specify up to four colors.
- SYSNAME – Limits events to those that are produced by this system. You can specify up to four SMF IDs. Note: If your OPSLOG contains events from only one system, this option does not limit events.
- User – Limits events to those that have matching data in the USER column of the OPSLOG. You can specify up to four user IDs.
- RULETRACE – Limits events to those that reflect data resulting from the RULETRACE parameter.
- A wildcard is a value that you specify that ends in an asterisk (*). The asterisk matches any one or more characters. You may specify wildcards for these options: Jobname, MSGID, Ruleset (rule set name and rule name), SYSNAME, and User.
- For example, if you specify IMS* as the value for the JOBNAME option, all events with job names that begin with the characters “IMS” appear in the OPSLOG Browse display.

Event-related OPSLOG Browse Profile Options

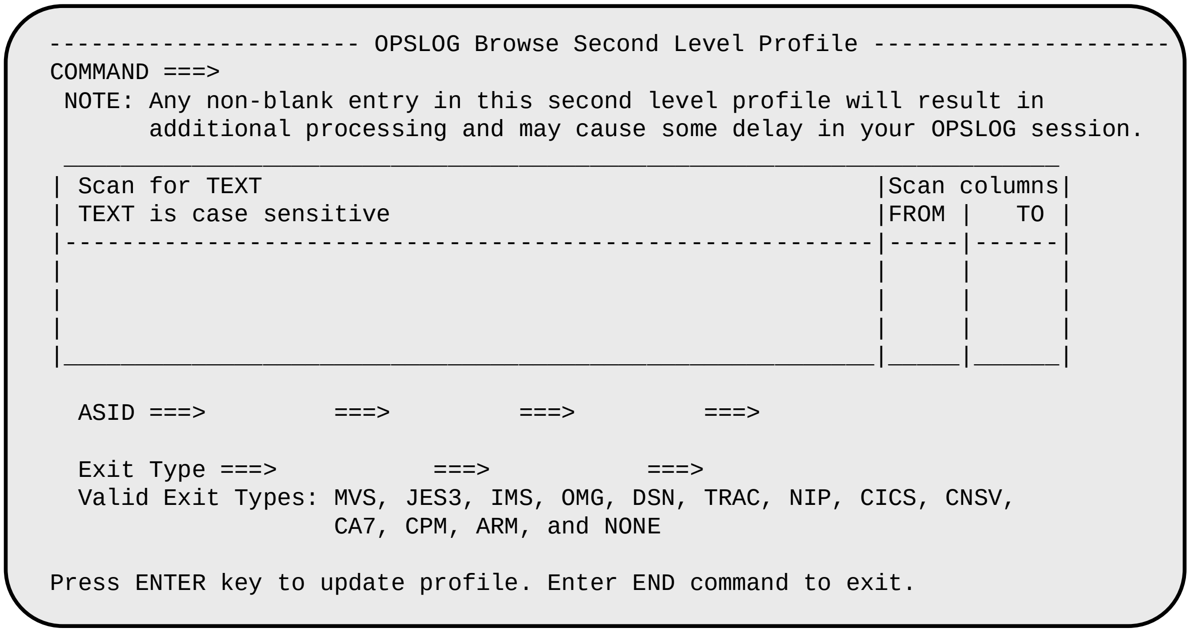
- MSG - CICS, CA-7, CPM, GDI, NIP, trace, WTO, WTOR, or WTL messages; or IMS messages, including those sent to MTO if the IOF is installed.
- CMD - MVS, JES, IMS, or other subsystem commands; VM operator commands.
- DIS -The disabling of rules.
- DOM - Delete-operator-message events.
- ENA - The enabling of rules.
- EOM - End-of-memory events.
- GLV - Global variable events.
- OMG - OMEGAMON exception events.
- REQ - Request events.
- SEC - Unicenter CA-OPS/MVS security events.
- TOD - Time-of-day events.
- SCR - EPI screen events.
- ARM - Automatic Restart Management events.
- EOS - End-of-step events.
- EOJ - End-of-job events.
- TLM - Time limit excession events.
- USS - UNIX System Services events.

Command-only OPSLOG Browse Profile Options

- CLEAR – Clears all of the nonevent-related profile entries and sets the event-related profile entries to their defaults. The PROFILE CLEAR or CLEAR command may be entered from the OPSLOG Browse Profile panel display.
- LIST – Lists all of the previously saved Profile IDs. The PROFILE LIST command may be entered in the Command field on the OPSLOG Browse Profile panel display.

Note: While in the OPSLOG Browse Profile panel display, you may enter a question mark (?) in the Profile ID entry field to display the Profile ID table.

OPSLOG Second Level Profile



```
--------------------- OPSLOG Browse Second Level Profile ---------------------
COMMAND ===>
 NOTE: Any non-blank entry in this second level profile will result in
       additional processing and may cause some delay in your OPSLOG session.
 ______________________________________________________________________
| Scan for TEXT                                           |Scan columns|
| TEXT is case sensitive                                  |FROM |   TO |
|---------------------------------------------------------|-----|------|
|                                                         |     |      |
|                                                         |     |      |
|                                                         |     |      |
|_________________________________________________________|_____|______|

  ASID ===>         ===>         ===>         ===>

  Exit Type ===>           ===>           ===>
  Valid Exit Types: MVS, JES3, IMS, OMG, DSN, TRAC, NIP, CICS, CNSV,
                    CA7, CPM, ARM, and NONE

Press ENTER key to update profile. Enter END command to exit.
```

2 - 12

OPSLOG Second Level Profile

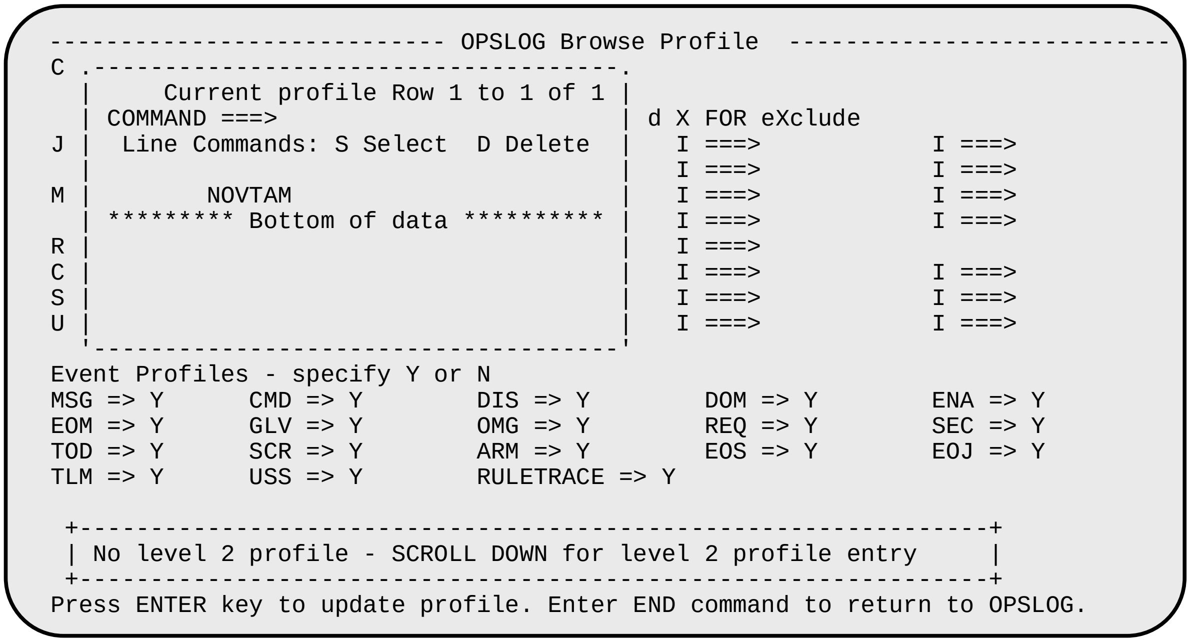
The Second Level Profile panel, shown above, allows you to filter on specific character strings in the message text. It may be displayed by scrolling down from the primary OPSLOG Browse Profile panel.

Note: The second level of filtering is costly in terms of CPU and storage resources and may significantly slow down your OPSLOG viewing response time.

Nonevent-related OPSLOG Browse Second Level Profile Options

- Scan for TEXT – Limits events to those that have this character string in its text. You can specify up to three character strings. Character strings are case-sensitive.
- ASID – Limits events to those that have this ASID. You can specify up to four ASIDs.
- Exit Type – Limits events to those that have this exit type. You may specify up to three exit types.

OPSLOG Current Profile ID



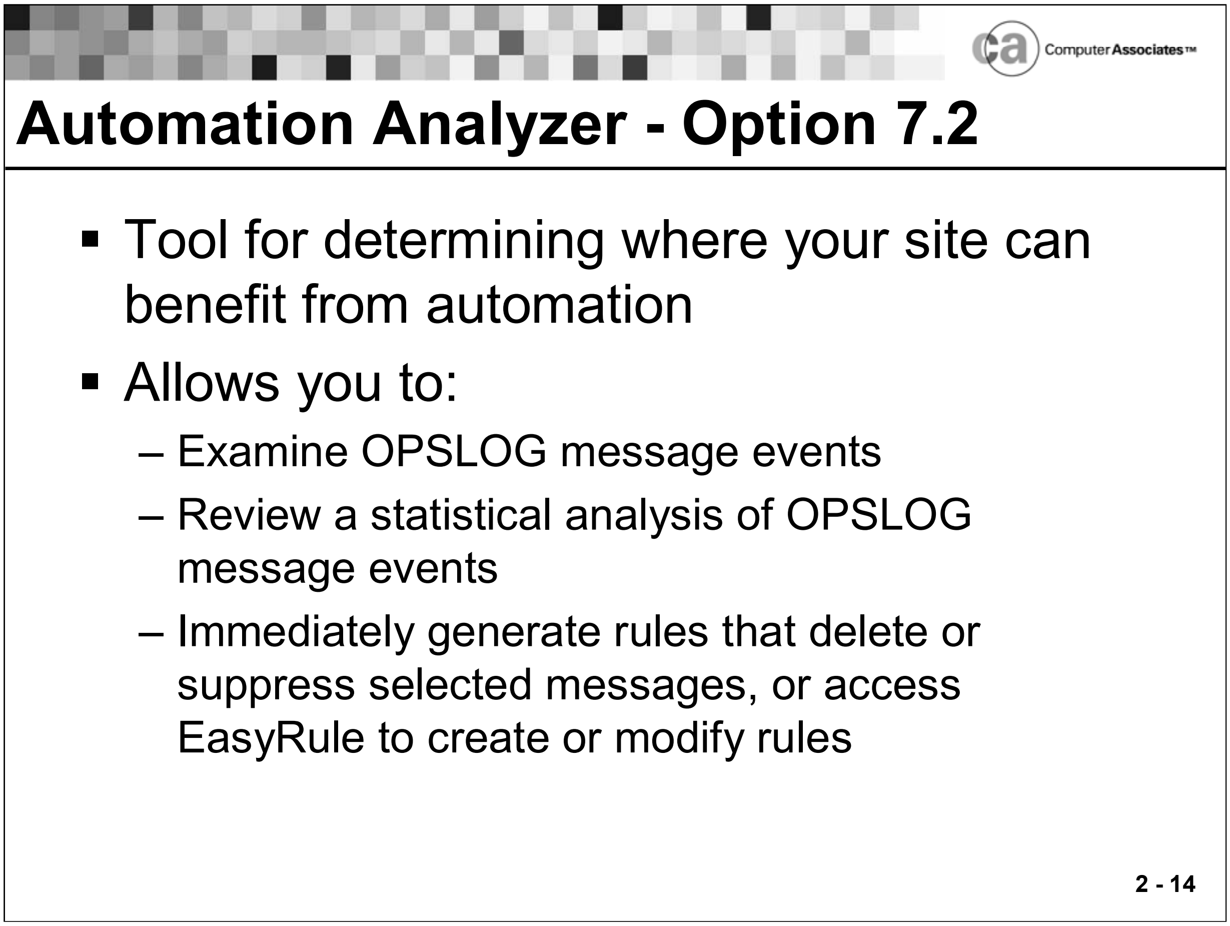
```
---------------------------- OPSLOG Browse Profile  ---------------------------
C .-------------------------------------.
  |     Current profile Row 1 to 1 of 1 |
  | COMMAND ===>                        | d X FOR eXclude
J |  Line Commands: S Select  D Delete  |   I ===>            I ===>
  |                                     |   I ===>            I ===>
M |        NOVTAM                       |   I ===>            I ===>
  | ********* Bottom of data ********** |   I ===>            I ===>
R |                                     |   I ===>
C |                                     |   I ===>            I ===>
S |                                     |   I ===>            I ===>
U |                                     |   I ===>            I ===>
  '-------------------------------------'
Event Profiles - specify Y or N
MSG => Y      CMD => Y        DIS => Y        DOM => Y        ENA => Y
EOM => Y      GLV => Y        OMG => Y        REQ => Y        SEC => Y
TOD => Y      SCR => Y        ARM => Y        EOS => Y        EOJ => Y
TLM => Y      USS => Y        RULETRACE => Y

 +-----------------------------------------------------------------+
 | No level 2 profile - SCROLL DOWN for level 2 profile entry      |
 +-----------------------------------------------------------------+
Press ENTER key to update profile. Enter END command to return to OPSLOG.
```

2 - 13

OPSLOG Current Profile ID

The OPSLOG Browse Current Profile ID panel, shown above, displays the different settings you have saved under individual IDs. A SAVE command entered with a valid profile ID creates a new entry into the table. An ID may be entered into the “Profile ID” entry field before issuing the SAVE command to create a new ID or overwrite an existing ID. If an ID is not entered into this entry field, OPSLOG prompts you for a new one.

This panel is displayed by entering a question mark (?) in the entry field “Profile ID” of the primary OPSLOG Browse Profile panel.



Automation Analyzer - Option 7.2

- Tool for determining where your site can benefit from automation
- Allows you to:
  - Examine OPSLOG message events
  - Review a statistical analysis of OPSLOG message events
  - Immediately generate rules that delete or suppress selected messages, or access EasyRule to create or modify rules

2 - 14

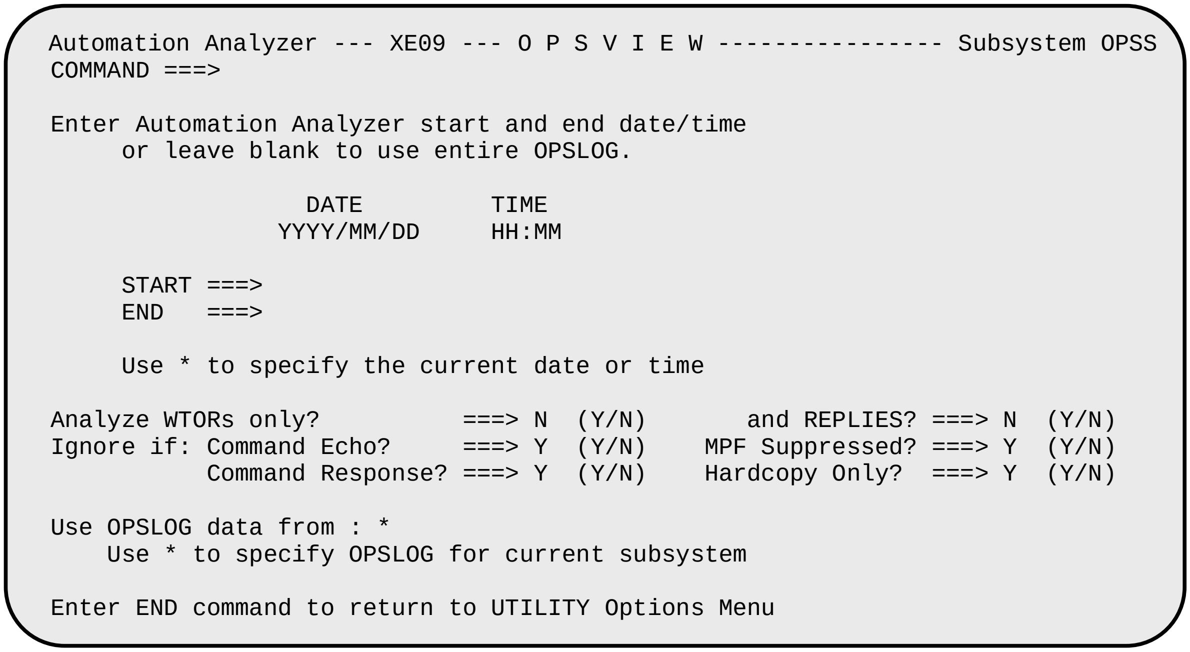
Automation Analyzer

The Automation Analyzer assists you in automating your site by showing you where your data center can benefit most from automation. With the information the Automation Analyzer provides, you are in a better position to decide whether automation of a message is desirable.

With the Automation Analyzer, you can:

- Examine message events that appear in OPSLOG.
- Review a statistical analysis of OPSLOG message events.
- Access an interface to Chicago-Soft's MVS/QuickRef product, through which you can view message descriptions.
- Immediately generate rules that delete or suppress selected messages, or access EasyRule to create or modify rules that take other actions.

Automation Analyzer (continued)



```
Automation Analyzer --- XE09 --- 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

Enter END command to return to UTILITY Options Menu
```

2 - 15

Automation Analyzer

Accessing the Automation Analyzer

- You can access the Automation Analyzer in either of the following ways:

  Type 2 in the Option field on the OPSVIEW Utilities Menu and press Enter.

  Use the ISPF jump function by typing =7.2 into any valid field within OPSVIEW and pressing Enter.

Fields on the Automation Analyzer Specification Panel

- Start

  To analyze only a portion of the message events, specify the desired starting date and time. All message events that occurred after the start date and time are analyzed.

  To specify the current date and time, type an asterisk (*) into the fields.

  To specify the beginning of the current day, type an asterisk (*) into the DATE field.

  If you leave the fields blank, the start date and time default to the date and time of the oldest record in the OPSLOG.

- End

  To analyze only a portion of the message events, specify the desired ending date and time. All message events that occurred before the end date and time are analyzed.

  To specify the current date and time, type an asterisk (*) into the fields.

  If you leave the fields blank, the end date defaults to the current date and the end time defaults to 23:59.

- Analyze WTORs Only

  Specify Y to restrict the analysis to only WTORs.

- Replies

  Specify Y if you want the analysis to include replies issued to WTORs. (If you specify Y in this field, you must also specify Y in the Analyze WTORs Only field.)

- Command Echo

  Specify N if you want the analysis to exclude echoes of commands.

- MPF Suppressed

  Specify N if you want the analysis to exclude messages that MPF is suppressing.

- Command Response

  Specify N if you want the analysis to exclude command response messages.

- Hardcopy Only

  Specify N if you want the analysis to exclude “hardcopy only” message.

- Use OPSLOG Data From

  Specify the OPSLOG that you want the Automation Analyzer to analyze.

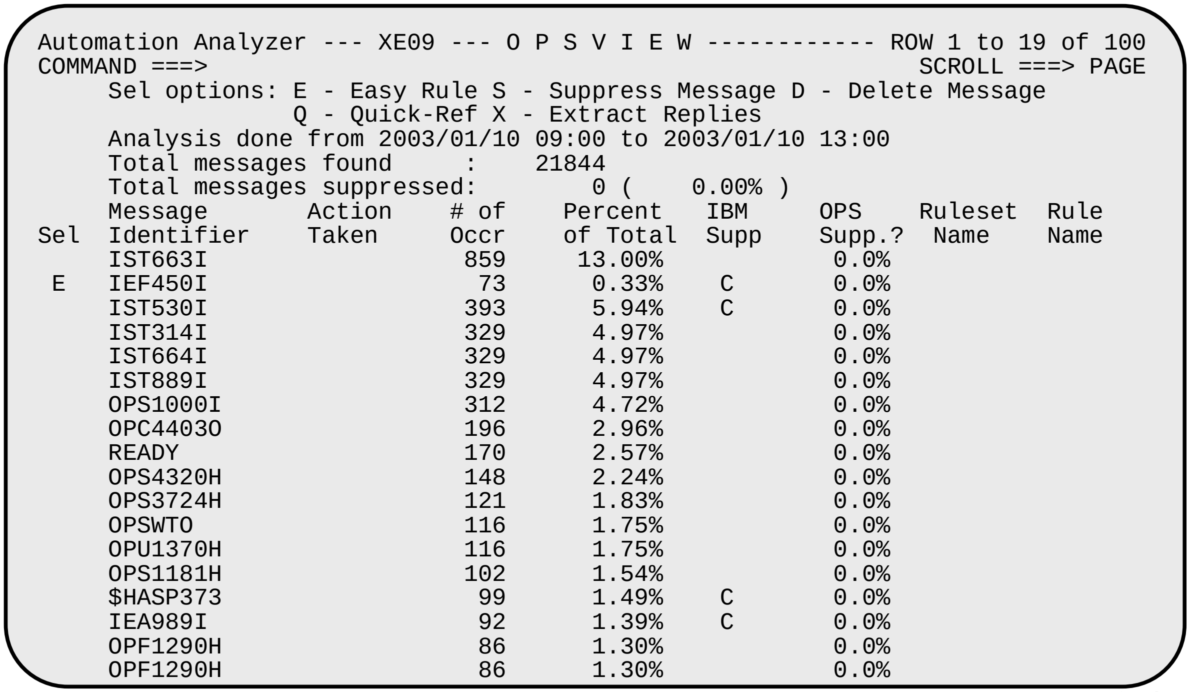
  If you enter a fully qualified data set name, enclose it within single quotes. If you omit the single quotes, OPS/MVS uses your TSO user ID as a prefix for the data set name.

  Specify an asterisk (*) if you want the Automation Analyzer to use the OPSLOG from the currently active subsystem.

Using the Automation Analyzer Specification Panel

- After you have indicated which messages you want to be analyzed on the Automation Analyzer Specification panel, press Enter. If you make no entries, the analysis starts from the beginning of the currently active OPSLOG and continues to the current time.
- A message appears to indicate that the analysis has begun. When the analysis is complete, the Automation Analyzer Results panel appears.

Automation Analyzer Results



```
Automation Analyzer --- XE09 --- O P S V I E W ----------- ROW 1 to 19 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 2003/01/10 09:00 to 2003/01/10 13:00
     Total messages found     :    21844
     Total messages suppressed:        0 (    0.00% )
     Message       Action    # of    Percent   IBM     OPS    Ruleset  Rule
Sel  Identifier    Taken     Occr    of Total  Supp    Supp.?  Name    Name
     IST663I                  859     13.00%            0.0%
 E   IEF450I                   73      0.33%    C       0.0%
     IST530I                  393      5.94%    C       0.0%
     IST314I                  329      4.97%            0.0%
     IST664I                  329      4.97%            0.0%
     IST889I                  329      4.97%            0.0%
     OPS1000I                 312      4.72%            0.0%
     OPC4403O                 196      2.96%            0.0%
     READY                    170      2.57%            0.0%
     OPS4320H                 148      2.24%            0.0%
     OPS3724H                 121      1.83%            0.0%
     OPSWTO                   116      1.75%            0.0%
     OPU1370H                 116      1.75%            0.0%
     OPS1181H                 102      1.54%            0.0%
     $HASP373                  99      1.49%    C       0.0%
     IEA989I                   92      1.39%    C       0.0%
     OPF1290H                  86      1.30%            0.0%
     OPF1290H                  86      1.30%            0.0%
```

2 - 17

Automation Analyzer Results

After the Automation Analyzer completes its analysis of the selected messages, a panel of results appears.

The summary statistics for the analysis appear in the top half of the panel. They include this information:

- The portion of the OPSLOG that was analyzed.
- The total number of unique message IDs that the Automation Analyzer found.
- The total number of messages in the analysis that were suppressed.
- The percentage of the messages in the analysis that were suppressed.
- Fields on the Automation Analyzer Results Panel

The following describes the fields on the Automation Analyzer Results panel:

- Sel

  Specify the action to take for a message. Values are D, E, Q, R, S, and X. See below for details about these options.

- Message Identifier

  The message ID.

- Action Taken

  The outcome of a previously entered Sel option.

- # of Occr

  The number of times the message ID appeared in the analyzed OPSLOG.

- Percent of Total

  The frequency with which the message ID appeared.

- IBM Supp

  A value indicating whether the message appears on IBM's conservative (C) list for message suppression, aggressive (A) list, or neither (blank). For more information about the list, refer to IBM's documentation.

- OPS Supp.?

  The percentage of times the message was suppressed by Unicenter CA-OPS/MVS.

- Ruleset Name/Rule Name

  The name of at least one rule that processes the message, and the name of the rule set to which the rule belongs.

Primary Commands for the Automation Analyzer Results Panel

- Locate msgid

  Scrolls the panel so that the line referring to msgid is the top line on the panel. You can specify a partial ID. For example, issue this command to locate the first message ID that begins with the characters IST: **`L IST`**

- REPORT

  Sends a report to your ISPF LIST data set. With the exception of the information that appears in the Action Taken field, the report includes all of the information that the Automation Analyzer Results panel provides.

- SORT columnname

  Sorts the list of messages according to the specified column. Values for columnname are: msg, msgid, and message (to sort by Message Identifier field); cnt, count, num, number, occ, and occurrence (to sort by # of Occr field); and sup, supp, and suppressed (to sort by OPS Supp.? field).

  Specify up to three values for columnname. For example, issue this command to sort the messages first by OPS/MVS suppression and then by message ID: **`SORT SUPP MSGID`**

Line Commands for the Automation Analyzer Results Panel

- D

  Immediately generates a rule that suppresses and deletes the message from SYSLOG. (Occurrences of a message that has been deleted will still be recorded in the OPSLOG.)

- E

  Invokes EasyRule so you can create or edit a rule for the message. Use this command if you want to create a rule that does something other than message suppression or deletion. This command is synonymous with the R line command. (The EasyRule feature is discussed next.)

- Q

  Lets you view the MVS/QuickRef product’s description of the message.

- R

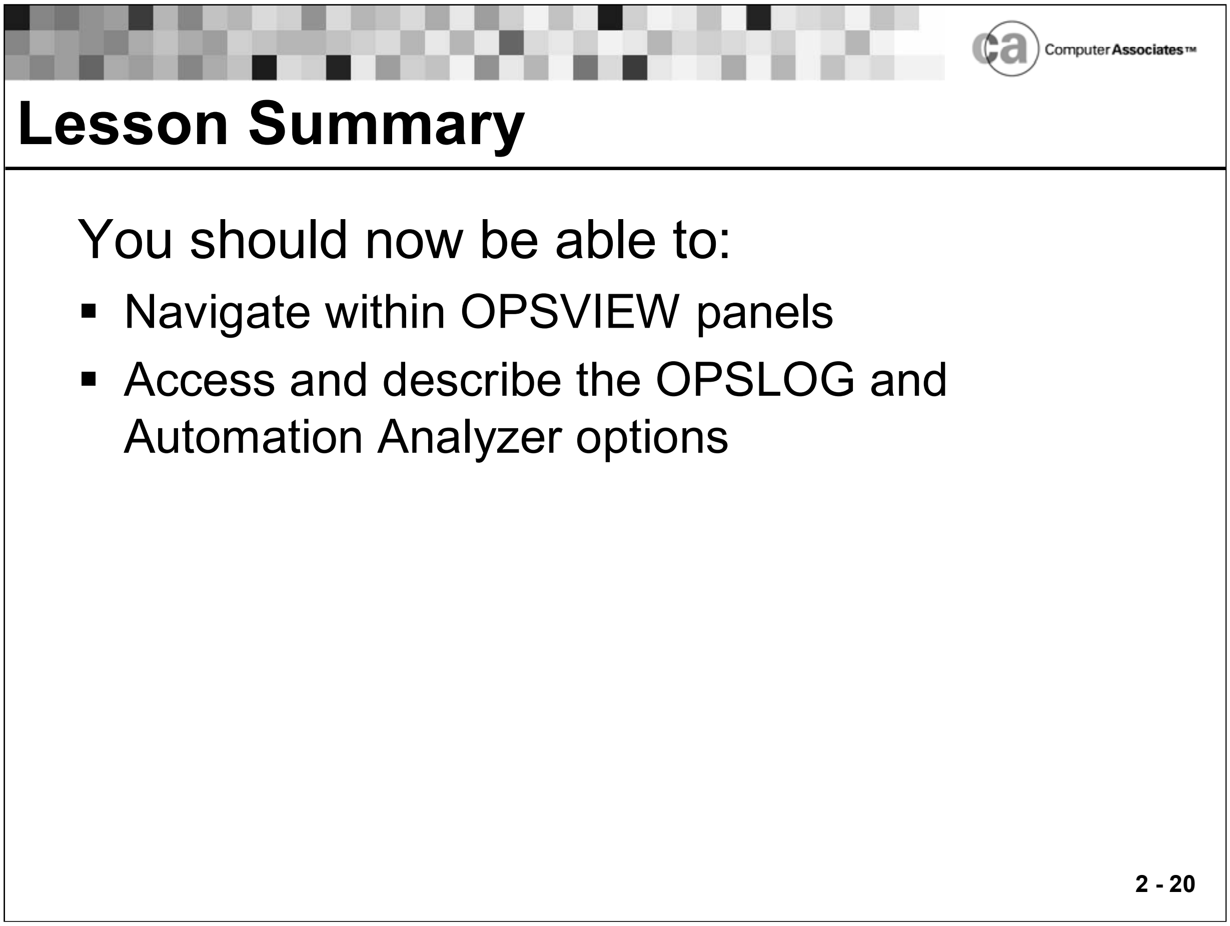
  Invokes EasyRule so you can create or edit a rule for the message. Use this command to create a rule that does something other than message suppression or deletion. This command is synonymous with the E line command. (The EasyRule feature is discussed next.)

- S

  Immediately generates a rule that suppresses the message.

- X

  Extracts the replies issued to a WTOR message for automation.



Lesson Summary

You should now be able to:

- Navigate within OPSVIEW panels
- Access and describe the OPSLOG and Automation Analyzer options

2 - 20

Lesson 2 Assessment

2 - 21

Lesson 2 Assessment

1. If you wanted to view all of the events that occurred in your z/OS system, which OPSVIEW option would you select?

____

2. What would you type to “jump” to the Automation Analyzer Specification panel?

____

3. Is there another way that you can access the Automation Analyzer Specification panel? If so, explain how.

____

4. Which command would you use to view a list of all possible data columns in the OPSLOG?

____

Lesson 2 Assessment (continued)

5. Which facility would you use if you wanted to generate rules but you had little or no REXX experience?

____

6. Where does the Automation Analyzer obtain the data that it uses in its analysis of the automation practices at your site?

____

7. Can you enter z/OS commands via OPSVIEW?

____

8. What would you type to “jump” to the EasyRule facility?

____

9. Can you invoke EasyRule from the Automation Analyzer?

____

10. Describe the difference between OPSVIEW and Unicenter CA-SYSVIEW.

____

Lesson 2 Activity

2 - 23

Lesson 2 Activity – OPSVIEW Practice

Using the user ID and password given to you by your instructor, sign on to the Education system and access Unicenter CA-OPS/MVS.

Using the OPSLOG Browse Feature

1. Enter 1 (OPSLOG) in the Option field on the OPSVIEW Primary Options Menu. The OPSLOG Browse panel displays.
2. Issue the DISPLAY command and select the following columns in the specified order:
   - 1 – Date
   - 2 – Time
   - 3 – JOBNM (job name)
   - 4 – MSGID (message ID)
   - 5 – MSGNO (message number)
3. Press PF3 and browse the OPSLOG, which now contains the columns you specified. (Be sure to scroll to the right.)

Lesson 2 Activity (continued)

4. Use the FIND command to position your cursor at occurrences of the following:
    - The word “abend”.
    - $HASP100 messages.
    - $HASP395 messages.
5. Use the LOCATE command to position your cursor at a message number of your choice.
6. Use the PROFILE command to filter the OPSLOG:
    - Exclude job names that begin with OPS (specify OPS*).
    - Exclude message IDs that begin with IEF (specify IEF*).
7. Issue an z/OS command (/ command). For example:
    - / D A,tsouserid

Using the Automation Analyzer

1. Enter 7.2 in the Option field on the OPSVIEW Primary Options Menu. The Automation Analyzer Specification panel appears.
2. Press Enter to analyze the entire OPSLOG. Unicenter CA-OPS/MVS displays the “OPSLOG analysis in progress” message. When the analysis is completed, Unicenter CA-OPS/MVS displays the Automation Analyzer Results panel.
3. Perform the following actions:
    - Extract a reply (X) for a message.
    - Delete a message (D) so that it will not appear in the SYSLOG.

    Notice that the text in the “Action Taken” column changes according to the action you perform.

Using the Command Feature

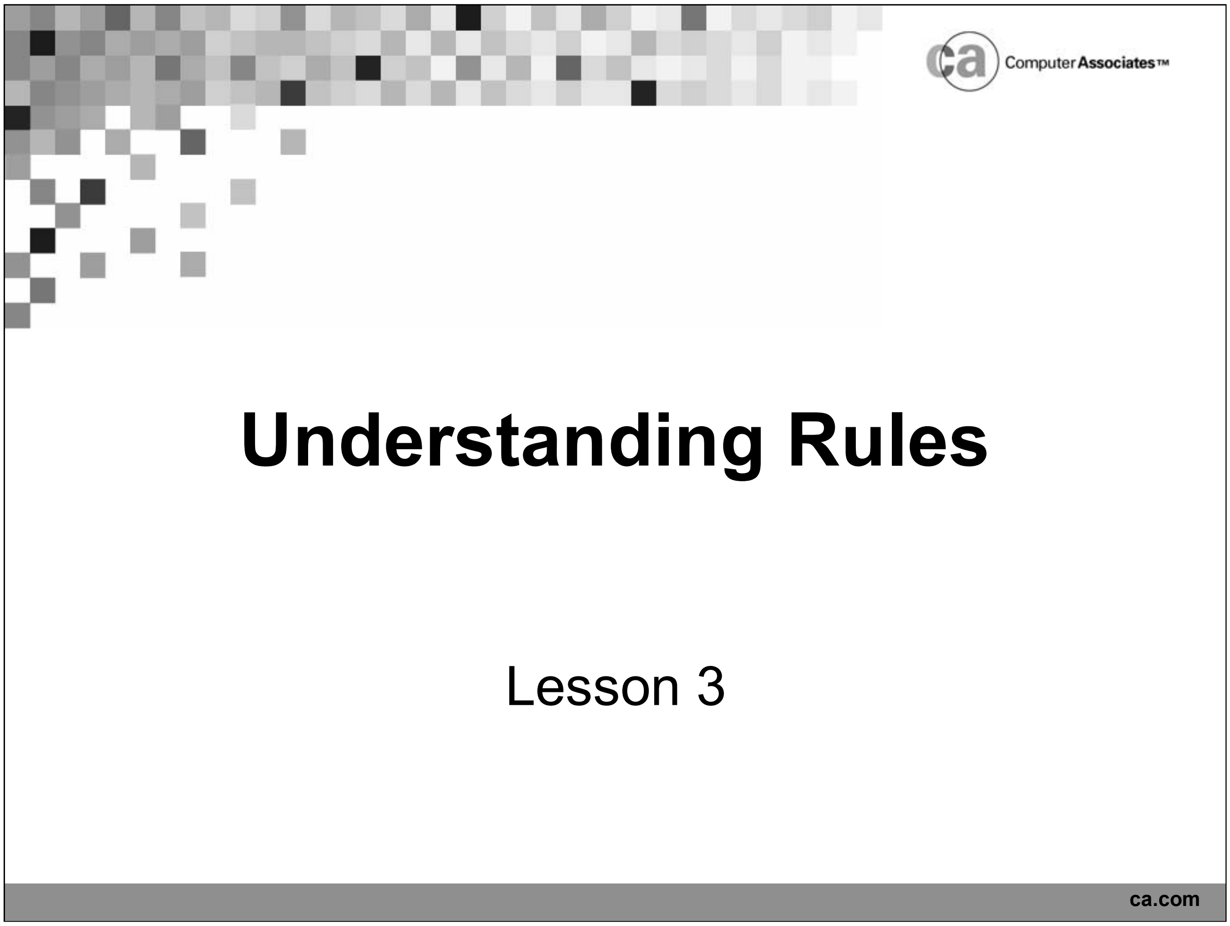
1. Enter 6 (Command) in the Option field on the OPSVIEW Primary Options Menu. The MVS/JES Command Processor panel displays.
2. Enter a display command (for example, D T) of your choice on the Command line.
3. Enter two or three more display commands on the Command line, one at a time.
4. Use the CMDLIST PAST command to view the commands you entered.
5. Place your cursor to the left of a command you want to execute and then press Enter.
6. Scroll through past commands using the UP2 and DOWN2 commands.

Lesson 2 Activity (continued)

Using the Tutorial

1. Enter T (Tutorial) in the Option field on the OPSVIEW Primary Options Menu. The Tutorial panel appears.
2. Enter 3 (Sys Control) in the Option field on the Tutorial panel.
3. Enter 1 (Address Space) in the Option field on the next Tutorial panel.
4. Explore the information regarding controlling address spaces.
5. Exit the Tutorial and then access the actual Address Space Control panel (OPSVIEW option 3.1). Examine the information that is presented on this panel.
6. Access the Tutorial again. The Tutorial panel displays.
7. Enter 4 (Control) in the Option field on the Tutorial panel.
8. Enter 1 (Parms) in the Option field on the next Tutorial panel.
9. Enter 1 (Parms) in the Option field on the next Tutorial panel.
10. Read about setting/displaying product parameters.

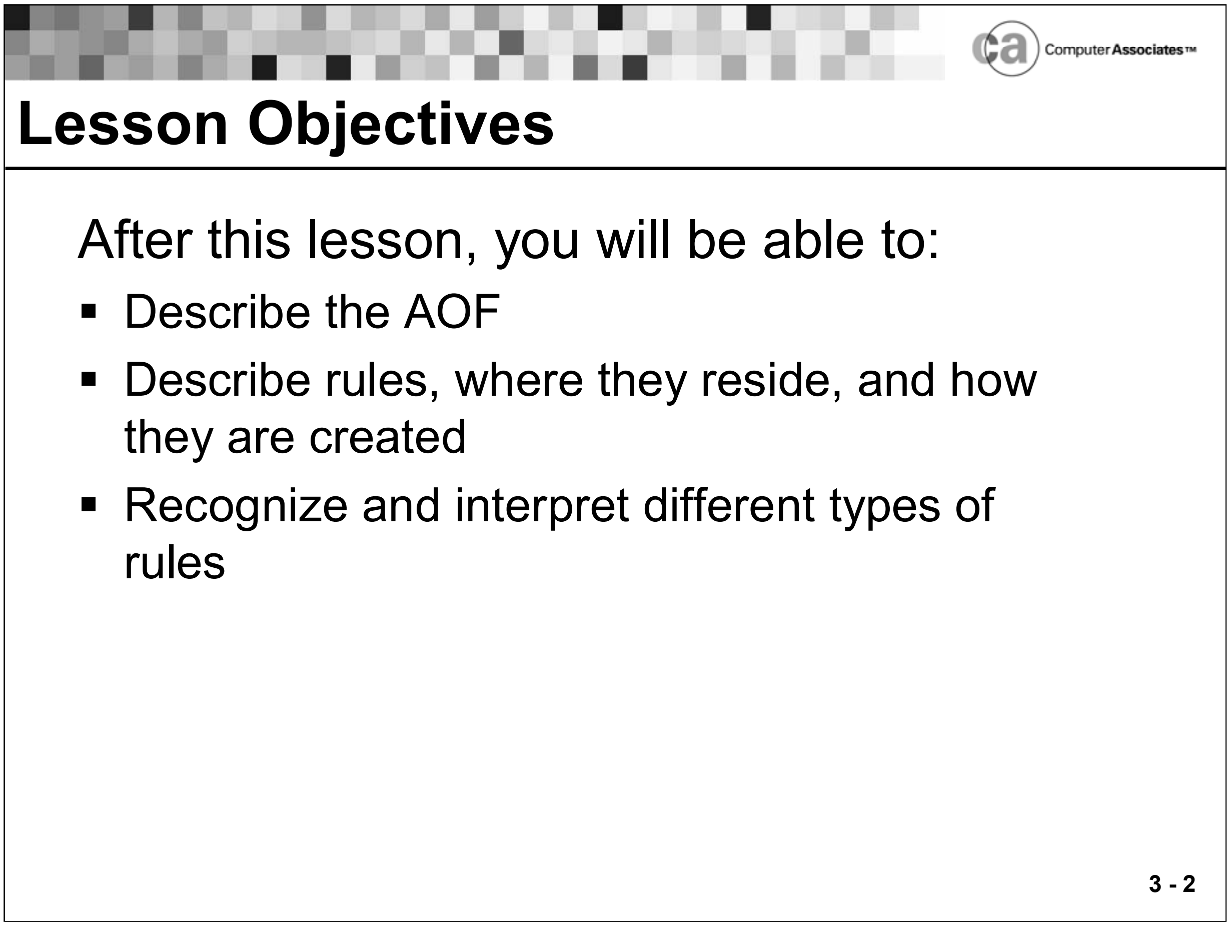
Notes:



Understanding Rules

Lesson 3

ca.com

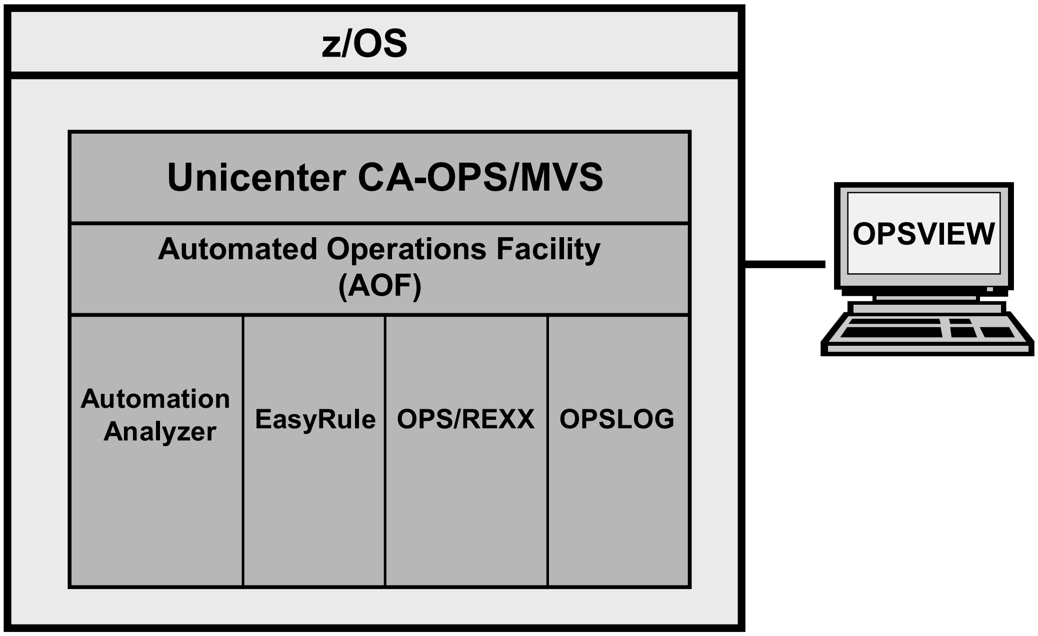


Lesson Objectives

After this lesson, you will be able to:

- Describe the AOF
- Describe rules, where they reside, and how they are created
- Recognize and interpret different types of rules

AOF



3 - 3

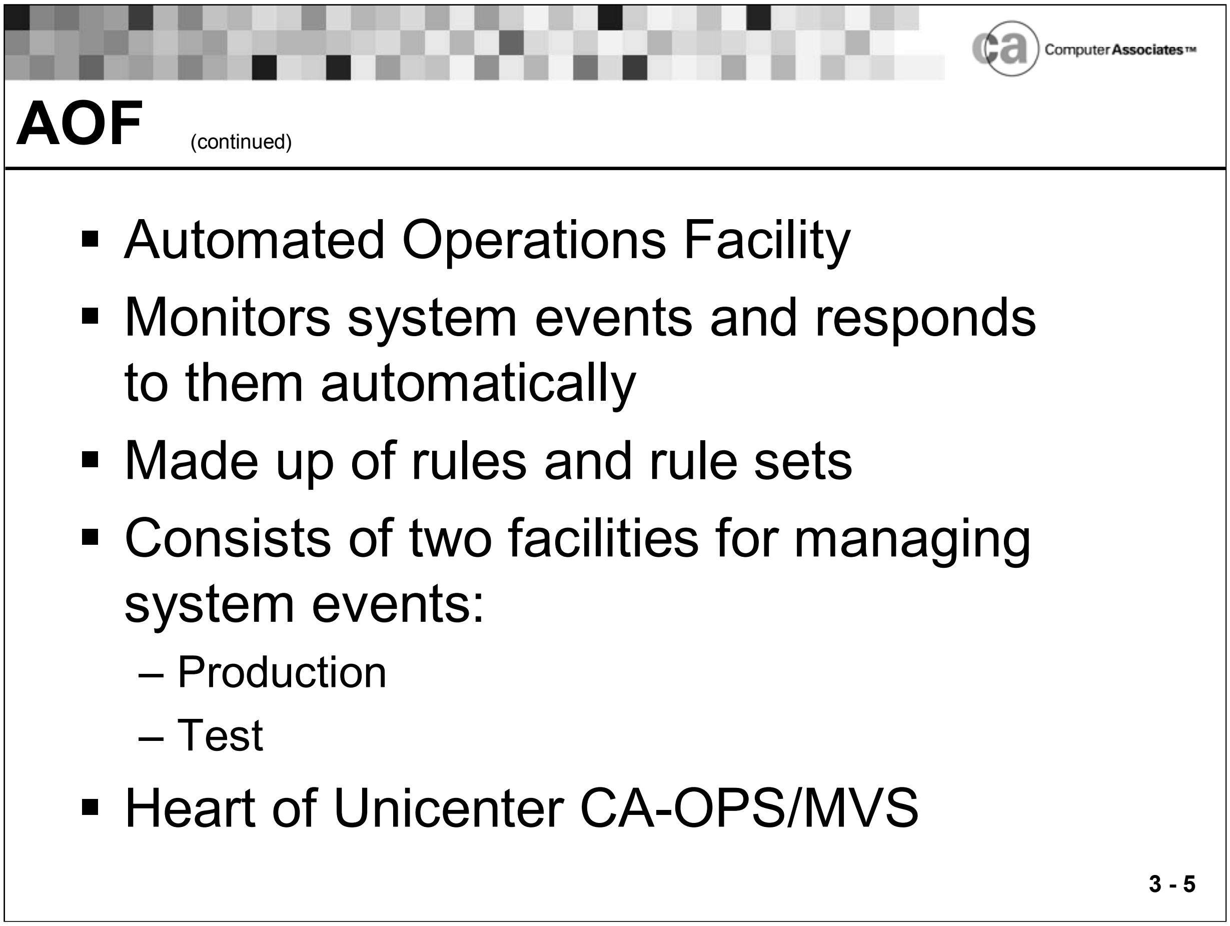
AOF

The ability to react to various system events is crucial when attempting to build effective automated applications. The Automated Operations Facility (AOF) is a component of Unicenter CA-OPS/MVS that monitors system events and responds to them automatically.

You determine the system events that the AOF recognizes—and how it responds to those events—by defining special OPS/REXX programs called AOF rules. AOF rules are classified as special OPS/REXX programs because they have a unique structure, reside in partitioned data sets (PDS) called rule sets, and are triggered by some system event.

The AOF can take action in response to the following types of system events:

- Automatic Restart Management events (ARM)
- Application Program Interface (API)
- Operator command events (CMD)
- Delete operator message events (DOM)
- End-of-job events (EOJ)
- End-of-memory events (EOM)
- End-of-step events (EOS)
- Global variable events (GLV)
- Message events (MSG)
- OMEGAMON exception events (OMG)
- End-user request events (REQ)
- Screen events (SCR)
- Security events (SEC)
- Time limit exceeded events (TLM)
- Time of Day events (TOD)
- UNIX System Services events (USS)



AOF (continued)

- Automated Operations Facility
- Monitors system events and responds to them automatically
- Made up of rules and rule sets
- Consists of two facilities for managing system events:
  - Production
  - Test
- Heart of Unicenter CA-OPS/MVS

3 - 5

AOF

The AOF is made up of rules and rule sets. A rule is a collection of OPS/REXX programs. A rule set is a collection of rules.

The AOF is the heart of Unicenter CA-OPS/MVS. It is where the majority of the work is performed.

The AOF consists of two separate facilities for managing system events:

- Production facility.
- Test facility.

AOF Production Facility

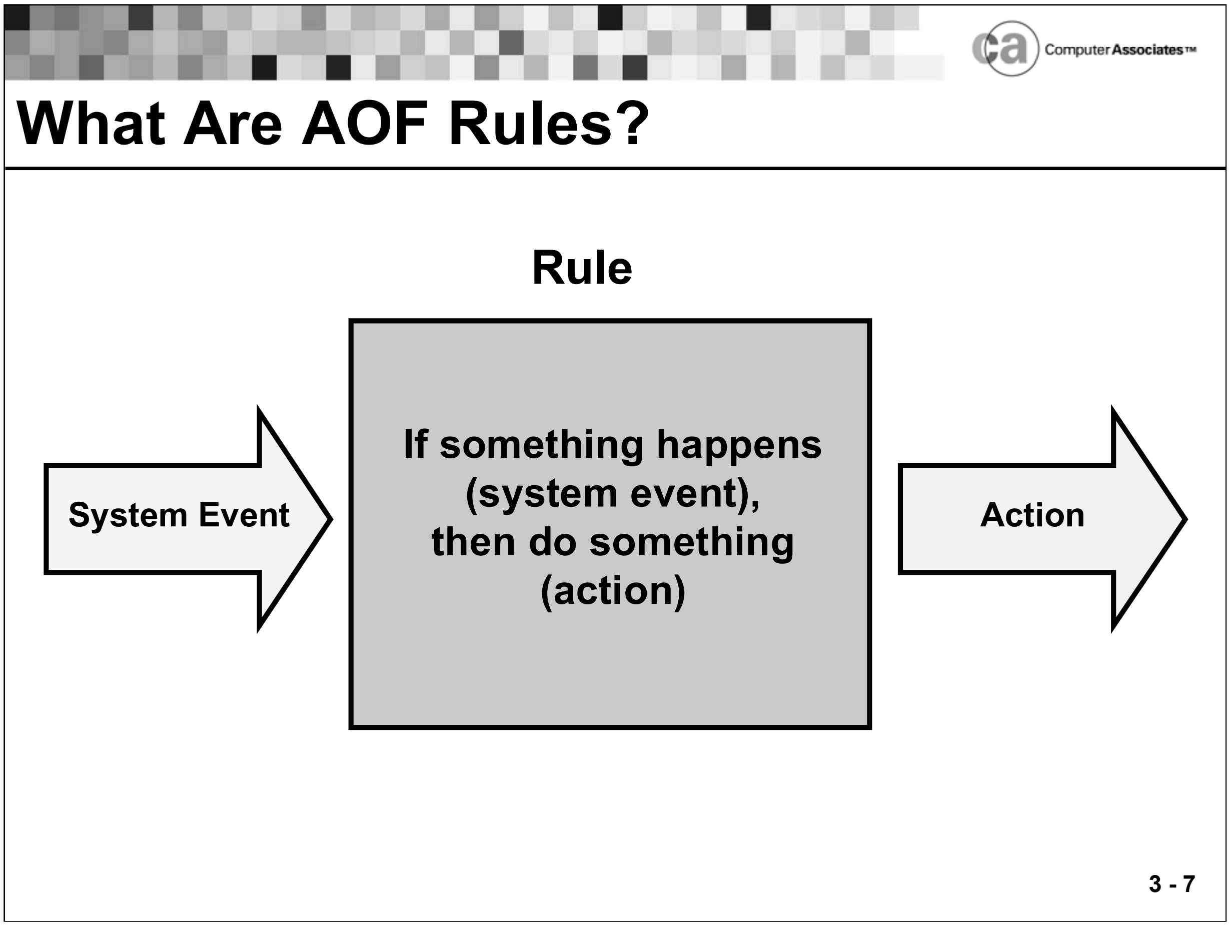
The production facility handles actual system events; for example, the issuing of a WTO message. It spans several Unicenter CA-OPS/MVS components:

- The Unicenter CA-OPS/MVS main address space (OPSMAIN), which provides services for detecting events and managing rules.
- OPSVIEW, which provides interactive applications that you can use to control AOF parameters (OPSVIEW option 4.1.1); control AOF rules and the AOF production compiled rules library (OPSVIEW options 4.5.1 and 4.5.2); and modify global variables (OPSVIEW option 4.8).
- The OPS/REXX language, which not only serves as the language in which you write rules, but also provides facilities for controlling the AOF.
- The Programmable Operations Interface (POI), which contains the OPSPARM command processor; the OPSPARM command processor sets several parameters that affect the overall operation of the AOF.

AOF Test Facility

The test facility lets you develop and test automation rules offline, before putting them into production.

The primary components of the test facility are OPSVIEW options 2.1 (for editing and testing AOF rules) and 2.2 (for maintaining the AOF test compiled rules library).



What Are AOF Rules?

Rule

System Event

If something happens (system event), then do something (action)

Action

3 - 7

What Are AOF Rules?

A rule is a collection of OPS/REXX programs. Each of these programs resides in special sections of a rule. Each section defines an aspect of the rule's execution.

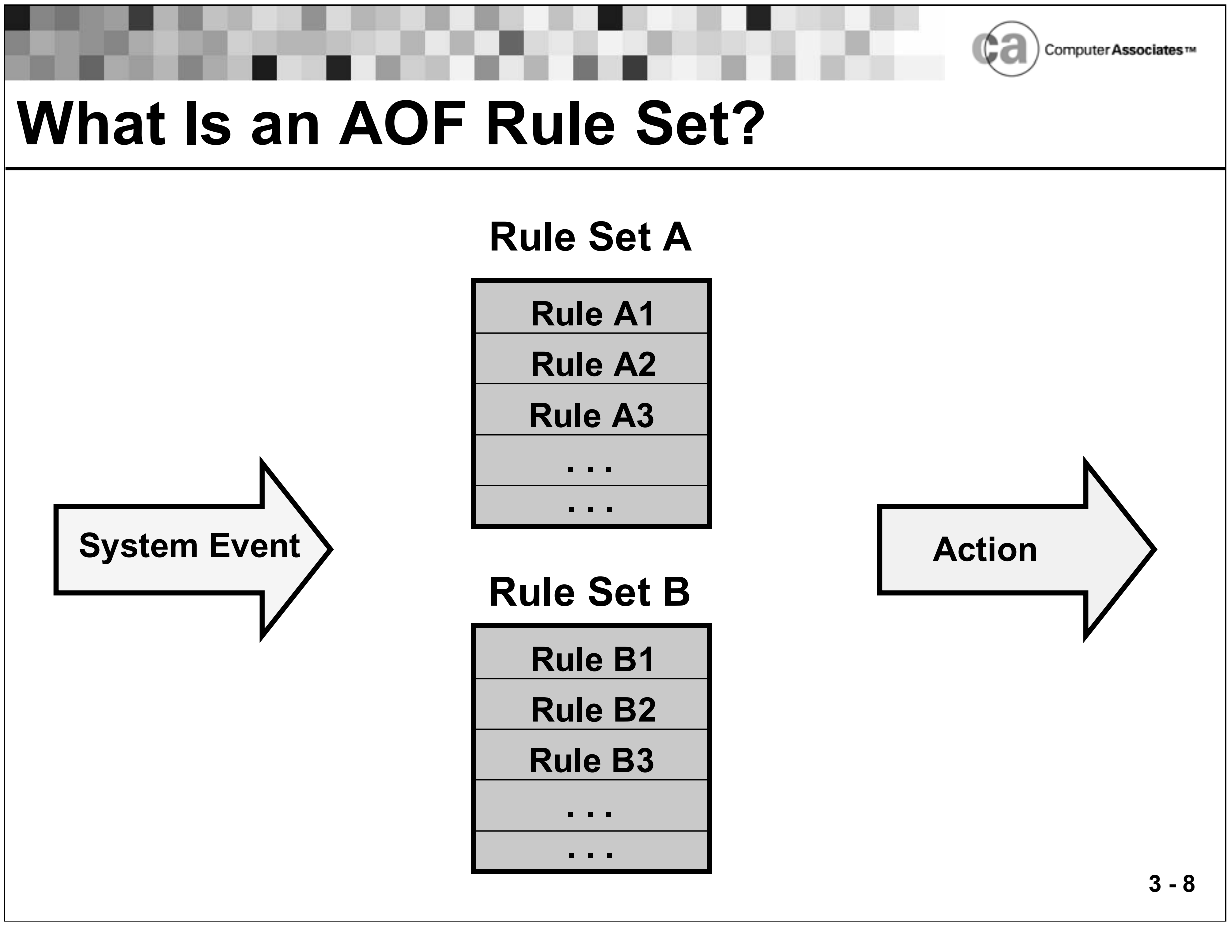
The rule sections are:

- Event definition (required).
- Initialization.
- Processing.
- Termination.

A rule must contain an event definition section and at least one other section. In addition, a rule may contain an END statement.

Rules are stored within rule sets.

Rules support automated operations within Unicenter CA-OPS/MVS.



What Is an AOF Rule Set?

System Event

Rule Set A

Rule A1

Rule A2

Rule A3

. . .

. . .

Rule Set B

Rule B1

Rule B2

Rule B3

. . .

. . .

Action

3 - 8

What Is an AOF Rule Set?

A rule set is a collection of rules. Rules are created and stored within rule sets. Rule sets are partitioned data sets (PDS). The initialization parameters named RULEPREFIX and RULESUFFIX determine the naming standard for rule sets.

Example:

- Using the following parameter settings:

  RULEPREFIX='OPSMVS.AOF'   RULESUFFIX='RULES'
- The rule sets would have a name mask of 'OPSMVS.AOF.*.RULES' where * is the rule set name. For example:

  OPSMVS.AOF.SUPPRESS.RULES
- Each rule would then occupy a separate member within the rule set:

  OPSMVS.AOF.SUPPRESS.RULES($HASP100)

In the example above, the rule named $HASP100 resides in the SUPPRESS rule set.

Determining Rule Set Names

Choose a naming convention for your rule sets that best fits your site's requirements. Here are some conventions to keep in mind when naming your rule sets:

- You can name rule sets according to rule type. For example, MSG for message rules, CMD for command rules, or TOD for time-of-day rules.
- You can name rule sets according to a particular automated application or request. For example, SUPPRESS for suppression rules, JES for JES-related rules, or IPLTIME for IPL-related rules.
- You can name rule sets according to individual groups or divisions within your organization. For example, CICSGRP for CICS personnel, OPERATNS for operations personnel, or IMSGRP for IMS personnel.

Multiple rule sets allow various groups using Unicenter CA-OPS/MVS within your data center to work independently of each other. Because each rule set is a separate data set, your security product can restrict rule set access to specific groups.

Defining Rule Sets

Follow standard ISPF/PDF data set naming requirements when creating rule sets using any of these methods:

- Option 3.2 of ISPF/PDF.
- TSO's ALLOCATE command.
- Option 4.5.1 of OPSVIEW.

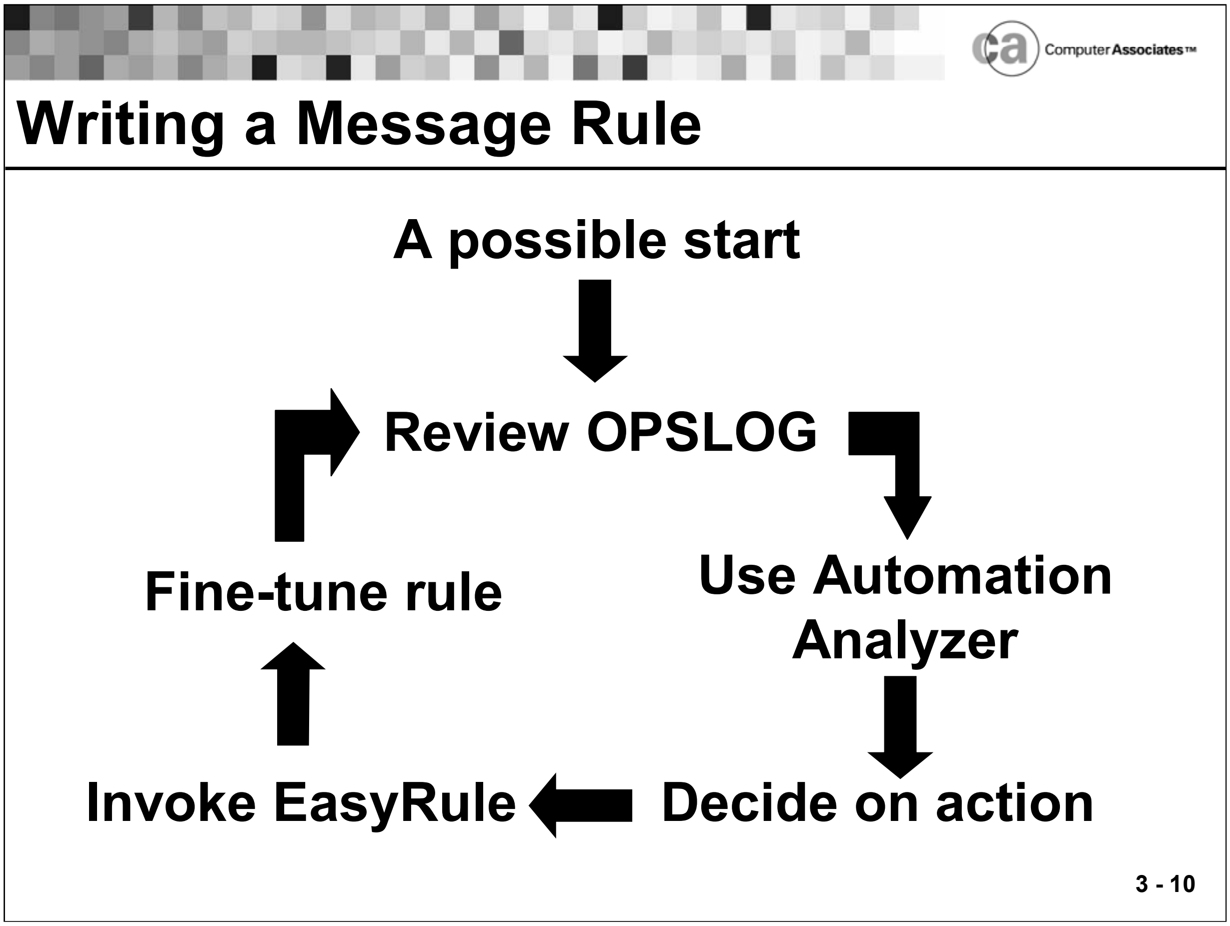
Specifying Parameters When Defining Rule Sets

Use these parameter values when allocating AOF rule sets:

- DSORG=PO
- RECFM=FB
- LRECL=80

Use these parameters at your discretion:

- DSN
- BLKSIZE
- SPACE
- UNIT



Writing a Message Rule

A possible start

Review OPSLOG

Use Automation Analyzer

Decide on action

Invoke EasyRule

Fine-tune rule

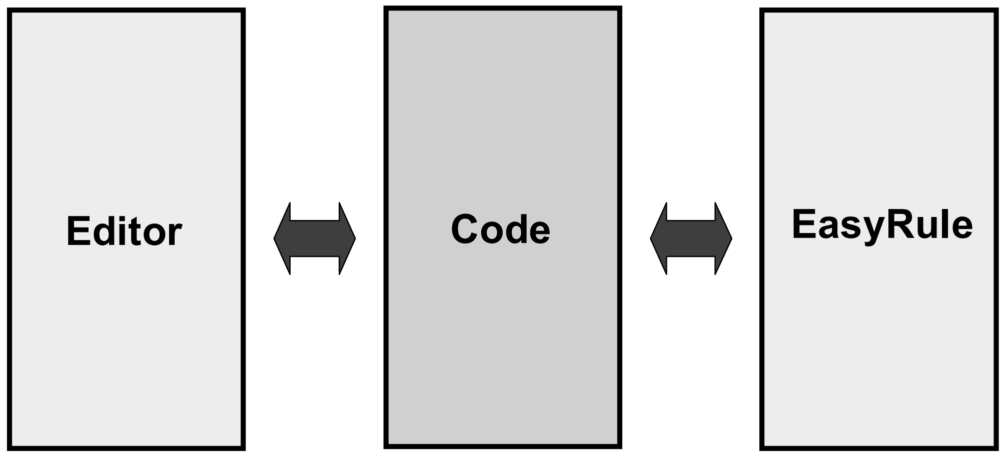
3 - 10

Writing a Message Rule

There are many factors to consider when you start to write a message rule. To get an idea of the types of messages for which you want to write rules, review the OPSLOG. The OPSLOG will give you a good idea of the types of messages you may want to suppress or have AOF take other actions on. The Automation Analyzer can also provide information about where you may benefit from automating procedures. Once you decide what types of actions you want to perform, you can then use EasyRule to write your rules.

Writing a Rule (continued)

Rule



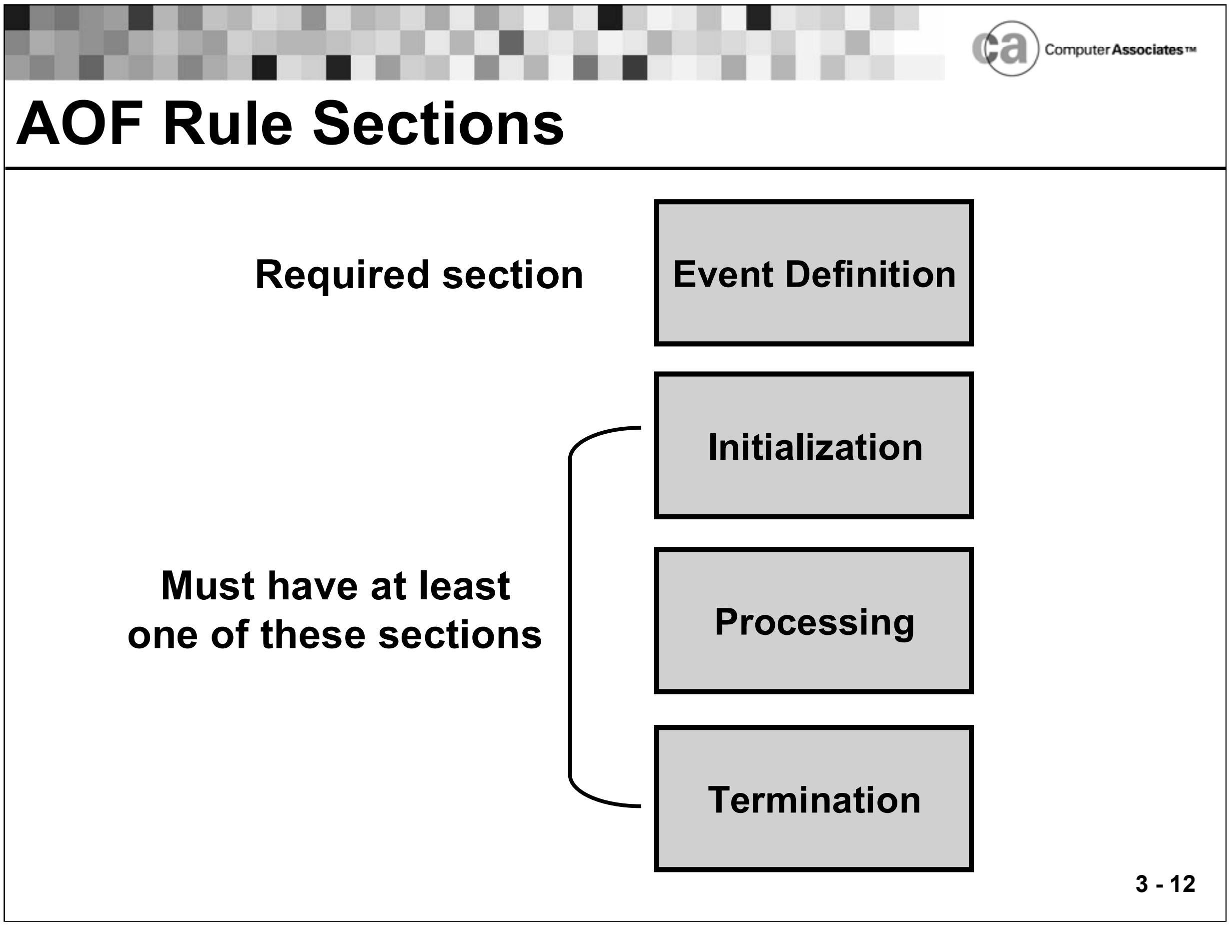
3 - 11

Writing a Rule

You can use any editor to write rules:

- Such as TSO EDIT or ISPF/PDF EDIT.
- OPSVIEW's AOF Edit options use a version of the ISPF/PDF editor, modified with special help panels.
- And you can use EasyRule to write your simple rules.

Unicenter CA-OPS/MVS provides example rules, located in the SYS1.OPS.RULES data set.



AOF Rule Sections

Required section

Event Definition

Must have at least one of these sections

Initialization

Processing

Termination

3 - 12

AOF Rule Sections

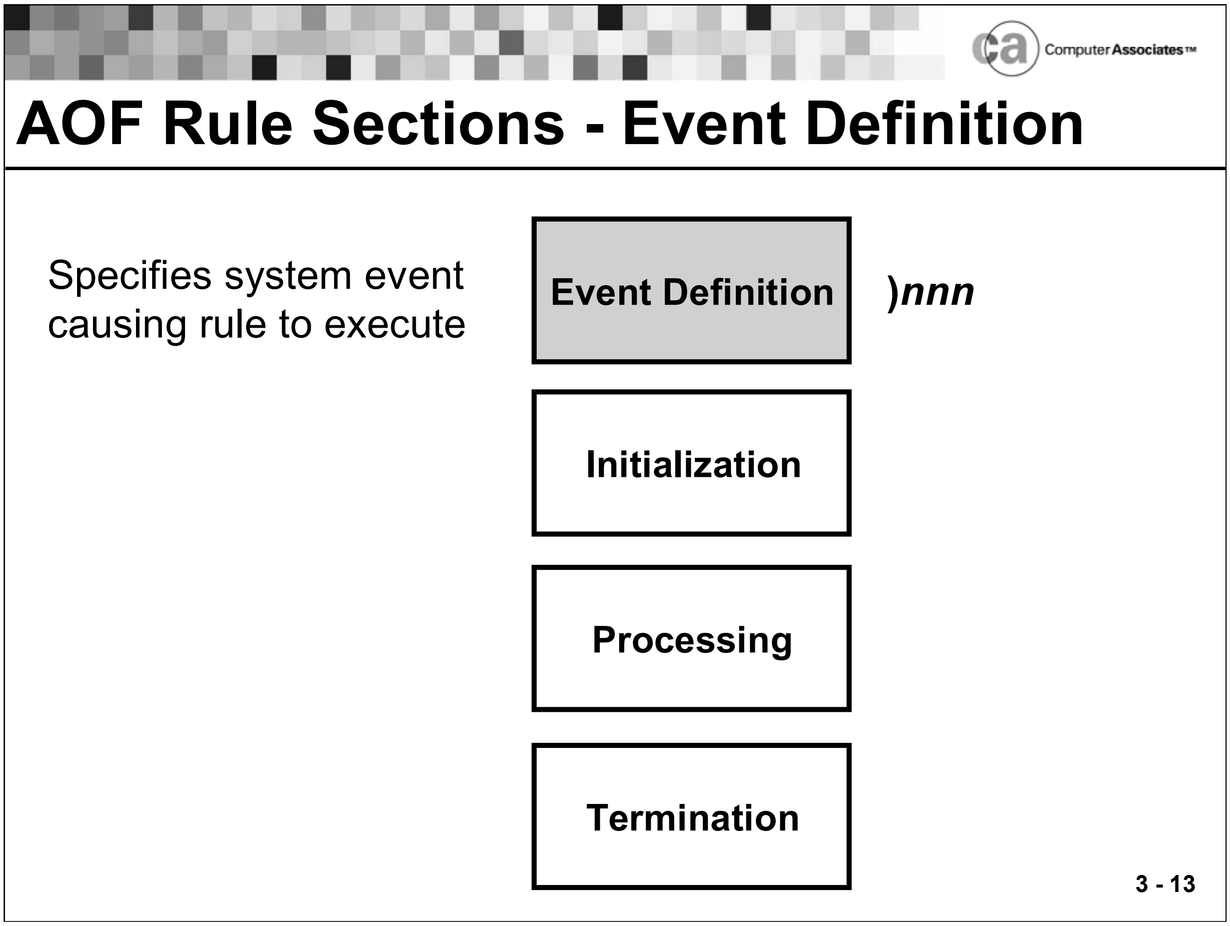
As you learned earlier, a rule contains sections. Each section defines an aspect of the rule's execution:

- Event definition (required).
- Initialization.
- Processing.
- Termination.

A rule must contain an event definition section and at least one other section. Each section is described in detail next.

Section headers delimit each section of a rule. Each section header must:

- Appear on a separate line by itself.
- Begin with a) character in column 1.



AOF Rule Sections - Event Definition

Specifies system event causing rule to execute

Event Definition
)nnn
Initialization
Processing
Termination

3 - 13

Event Definition

The event definition section of a rule specifies the system event that will cause the rule to execute. Unicenter CA-OPS/MVS uses the information in the event definition section to determine when to run the processing section of the rule. The event definition section is required and is always the first section of a rule.

Use this format for coding the event definition section:

```
)eventtype eventspec
```

The eventtype value is a three character designator identifying the AOF event type that the rule is intended to respond to.

Here are the valid three character designators:

)ARM)CMD)DOM)EOJ)EOM
)EOS)GLV)MSG)OMG)REQ
)SCR)SEC)TLM)TOD)USS

The eventspec value is a character string template that matches some event identifier (such as message IDs for MSG events, system command verbs for CMD events, time specifications for TOD events).

Note the following when specifying eventspec:

- The character string must match the event identifier exactly. For example, a value of "IEC205" matches an "IEC205" identifier only.
- You can use the wildcard character (*).
- Examples:

  IEC* matches IEC234, IECTL56, IEC67505, and any other event identifier containing an IEC prefix.

  IEC*05 matches IECD05, IEC205, IEC67505, and so on.

  *05 matches any identifier ending with 05.

  * alone matches any identifier.

Executing Multiple Rules in Response to a Single Event

You can write any number of rules that respond to a single event. Except for time rules with exactly matching event criteria, rules execute in a predictable order, as shown:

1. Rules with the most specific event criteria are tested first.

   - Message rules with most-specific to least-specific event specifiers are tested in this order:

     IST020I (seven significant characters—most specific)

     IST*I (four significant characters—less specific)

     IST* (three significant characters—least specific)

2. Event specifiers containing the same number of significant characters are tested in the order of longest prefix length.

   - If three event specifiers all contain six significant characters, they are tested in this order:

     IST02*I (five-character prefix)

     IST0*0I (four-character prefix)

     IST*20I (three-character prefix)

3. Rules containing identical event specifiers are tested in unpredictable order. If you want the rules to execute in a particular order, combine them into a single rule. (Do not combine unrelated rules, however, because doing so makes applications difficult to manage.)

Resolving Conflicting Rules

Conflicts may occur if more than one rule responds to a single event. For example, one rule may want to color a message white while another may want to color it red. In such cases, the following guidelines apply:

- In conflicts between two rules, the action specified by the rule which executed last will usually occur.
- To determine event disposition, the AOF uses the highest-precedence return value from the rule's processing section.

Coordinating Rules Processing for a Single Event

The AOF supports a special event-related variable that you can use to control how multiple rules execute when a single event occurs. When the first rule executes in response to an event, it can set the value of a special variable which other rules (triggered by the same event) can check or modify.

Each variable is eight bytes long and can contain any type of data. Name special variables according to this format:

- eventtype.USER

More About Event Types

This lesson focuses on command, message, time-of-day, and security rules.

- Command (CMD) Event

  A command event occurs when any z/OS or subsystem command is issued on the system.

- Message (MSG) Event

  A message event occurs when a system component sends a message to a console or a system log. The AOF recognizes and responds to these types of messages:

  z/OS

  IMS

  CICS (transient data queue messages)

  JES2/JES3

  Application-generated WTOs (write-to-operator), WTORs (write-to-operator-with-reply), and WTLs (write-to-log)

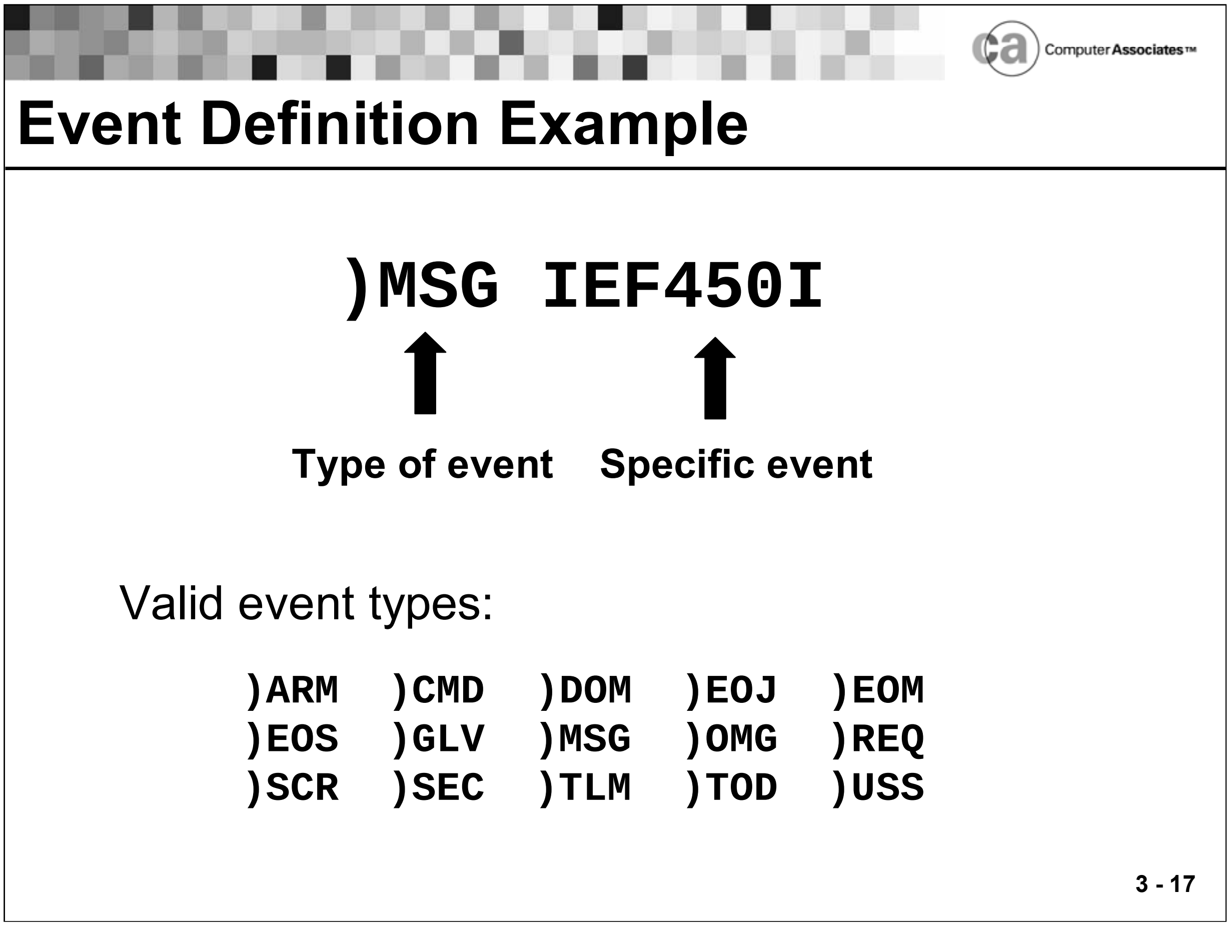
  Job log- or log file-directed

- Time-of-day (TOD) Event

  A time event can occur when automation needs to be performed at a specified time, date, or after a specified time interval.

- Security (SEC) Event

  A security event occurs when you invoke any Unicenter CA-OPS/MVS facility (for example, issuing the OPSCMD command processor to issue an z/OS command), allowing security for Unicenter CA-OPS/MVS to be performed by coding OPS/REXX programs rather than sophisticated assembler exits.



Event Definition Example

```
)MSG IEF450I
```

Type of event **Specific event**

Valid event types:

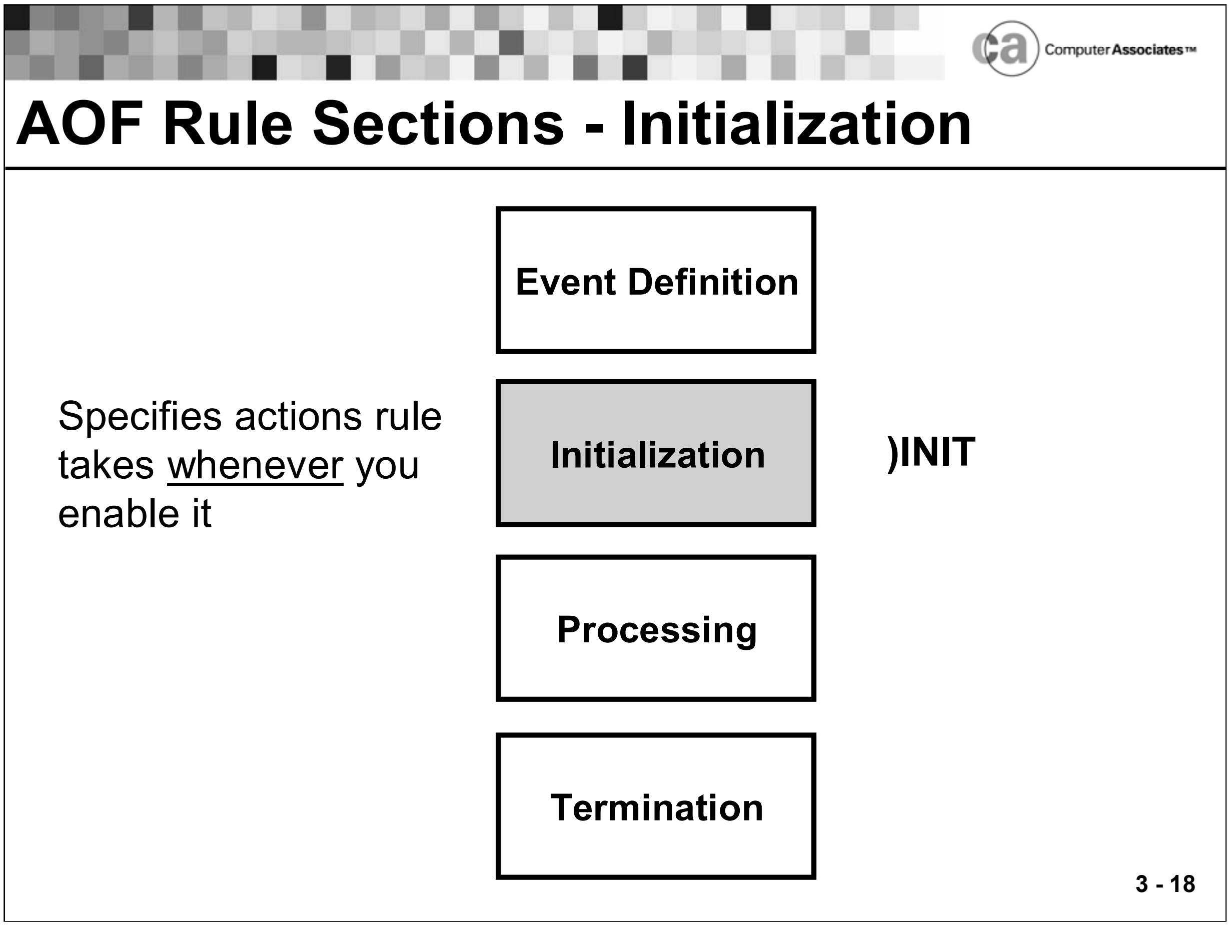
```
)ARM  )CMD  )DOM  )EOJ  )EOM
)EOS  )GLV  )MSG  )OMG  )REQ
)SCR  )SEC  )TLM  )TOD  )USS
```

3 - 17

Event Definition Example

In the following example, a rule is being defined for a message event. The specific event for which this rule is being defined is the IEF450I message.

```
)MSG IEF450I
```



AOF Rule Sections - Initialization

Specifies actions rule takes whenever you enable it

3 - 18

Initialization

The initialization section of a rule specifies actions the rule takes when it becomes enabled. The initialization section is optional. If included, it always follows the event definition section. This section executes in the Unicenter CA-OPS/MVS main address space.

Use this format for coding the initialization section:

```
)INIT
/* Insert initialization section action(s) */
```

The initialization section can contain OPS/REXX programs of varying complexity. Examples of some actions that you can program in the initialization section include:

- Issue operator commands (for z/OS, JES3, and VM).
- Issue TSO commands or CLISTs.
- Check the validity of the rule-enable request.
- Set initial values for variables.

Valid Return Values

- The OPS/REXX RETURN statement allows or disallows a rule's enablement.
- Valid values for a RETURN statement in a rule's initialization section are:

  ACCEPT

  Allows the rule to be enabled.

  REJECT

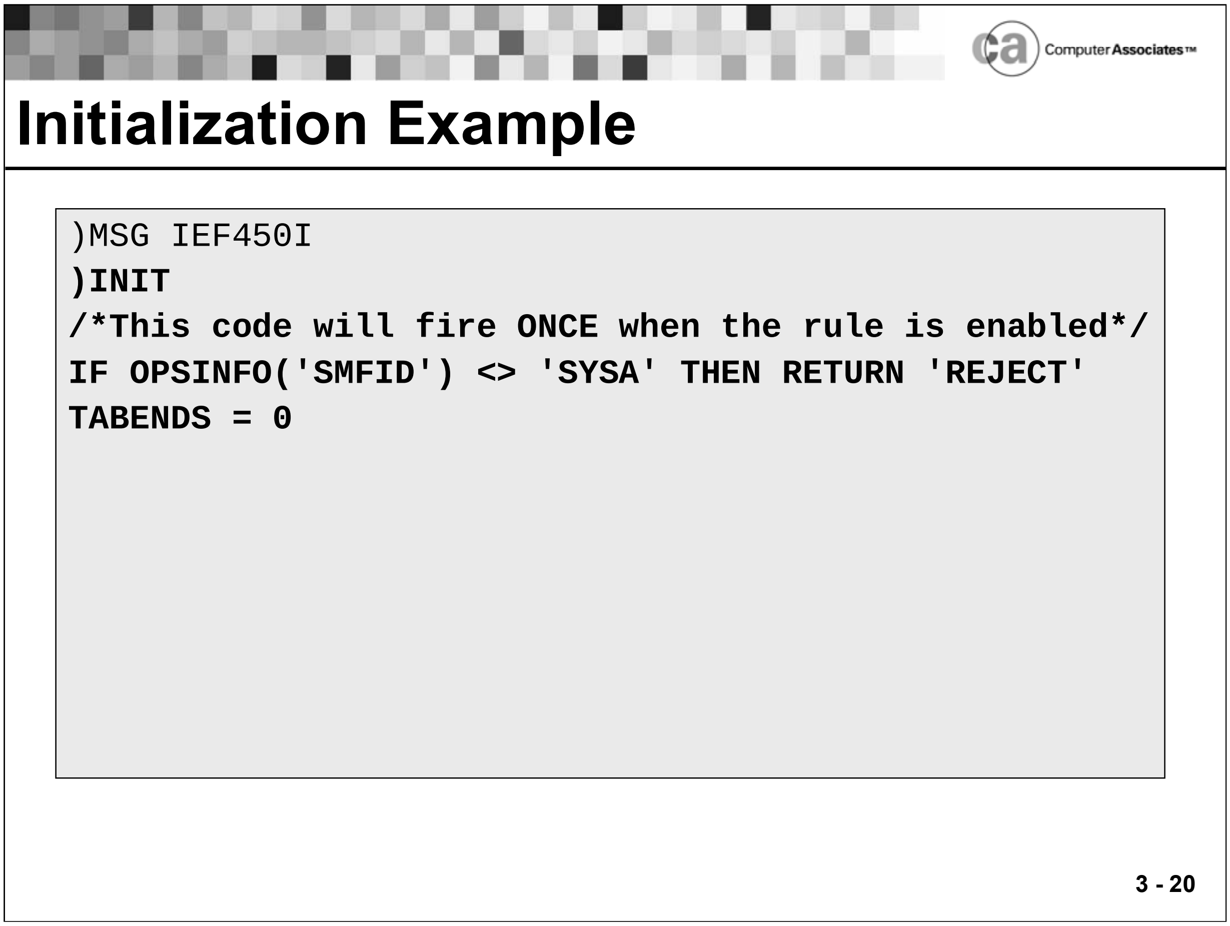
  Prevents the rule from being enabled.

  Note: The return values listed here are character constants, rather than keywords.

- Example. This statement prevents the rule from being enabled:

  ```
  RETURN "REJECT"
  ```

- If you do not specify a return value, the default return value is ACCEPT.
- If a run-time error occurs, the return value is ACCEPT (assuming that the error occurs while the initialization section is executing).



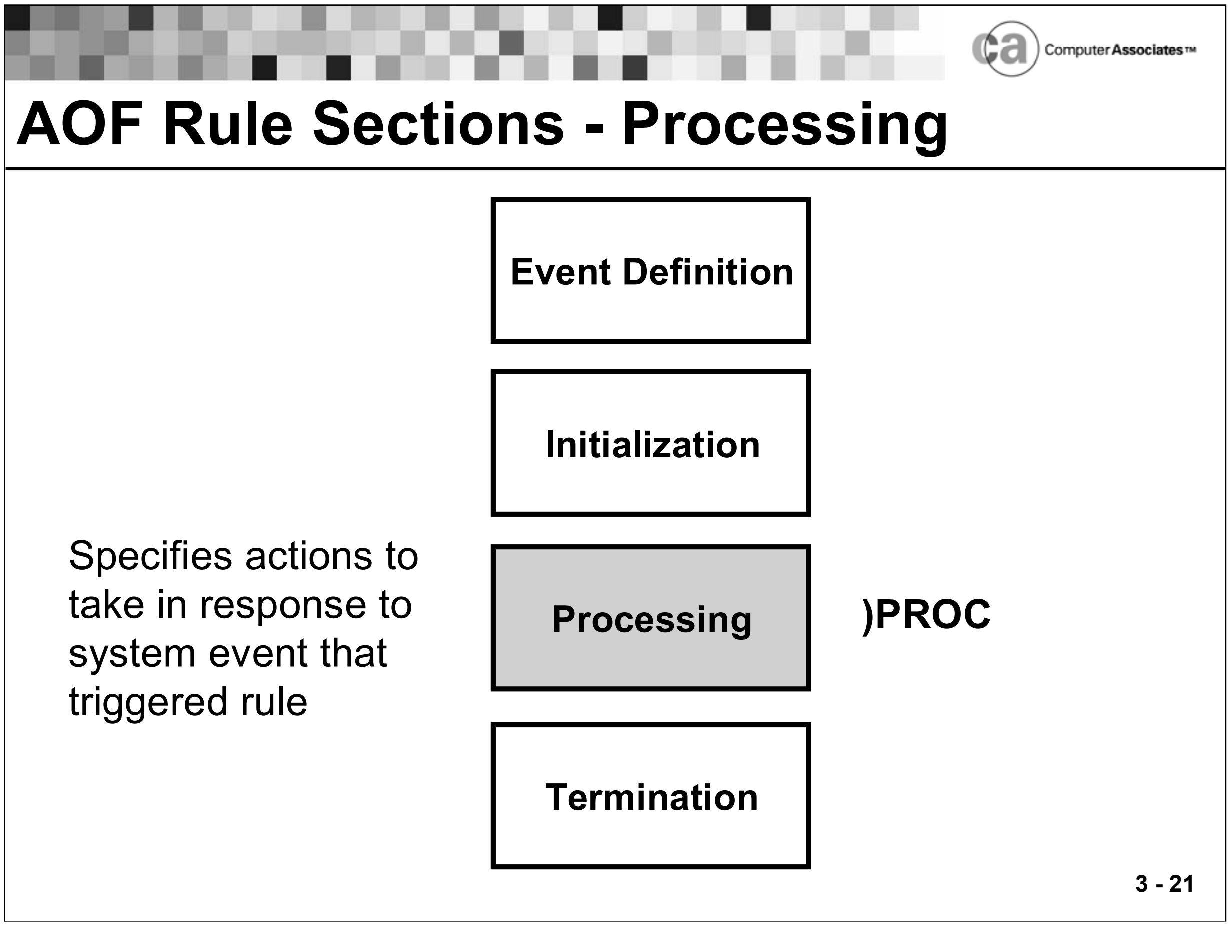
Initialization Example

```
)MSG IEF450I
)INIT
/*This code will fire ONCE when the rule is enabled*/
IF OPSINFO('SMFID') <> 'SYSA' THEN RETURN 'REJECT'
TABENDS = 0
```

3 - 20

Initialization Example

The example builds upon the event definition example. The OPS/REXX OPSINFO function is used to obtain the current SMFID and determine whether the rule will be enabled on system SYSA. The code rejects (that is, it does not allow) enablement on all other systems. The example code also initializes a static variable called TABENDS to zero. This variable is utilized in the processing section of the rule.



AOF Rule Sections - Processing

Specifies actions to take in response to system event that triggered rule

3 - 21

Processing

The processing section of a rule specifies the action(s) the rule initiates in response to the AOF detecting the system event that is defined in the event definition section of the rule. The processing section is optional. If included, it always follows the event definition section and initialization section (if it exists). This section runs synchronously in the address space where the system event occurred. No other processing can occur in that address space until AOF processing has completed.

Use this format for coding the processing section:

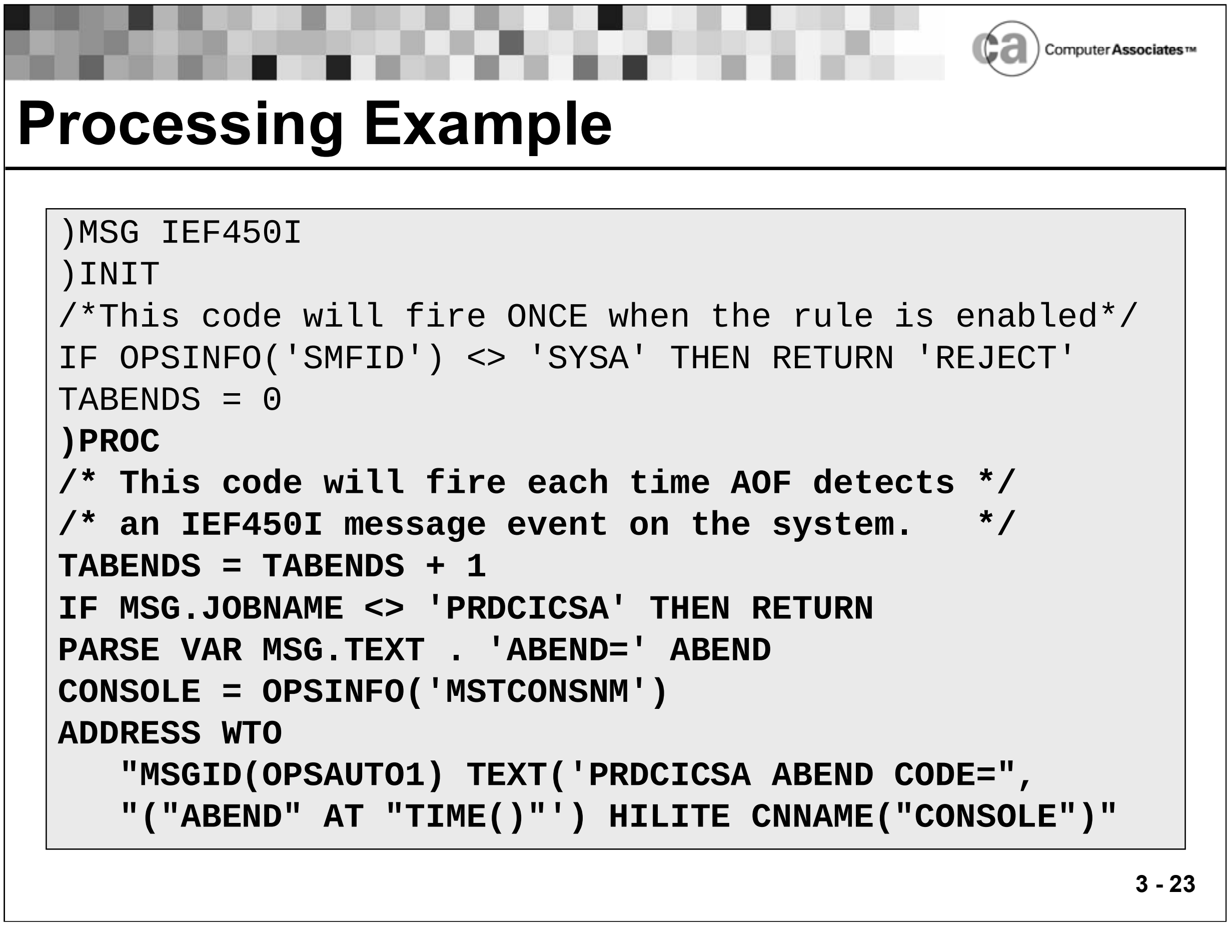
```
)PROC
/* Insert processing section action(s) */
               .
               .
               .
```

The processing section can contain a simple OPS/REXX statement or a complex OPS/REXX program. You can use the processing section to:

- Test for subsets of the event definition to find other possible system events that may be of interest.
- Collect information about the event and update global variables.
- Change, suppress, or route a message (when responding to a message event).
- Change or suppress a command (when responding to a command event).
- Reply to a WTOR message.
- Issue z/OS, JES3, or VM operator commands.
- Invoke TSO commands or CLISTs.

Valid Return Values

- The OPS/REXX RETURN statement works differently according to the type of event that the rule is acting upon. Refer to the documentation for detailed information on valid return statement values for each event type.



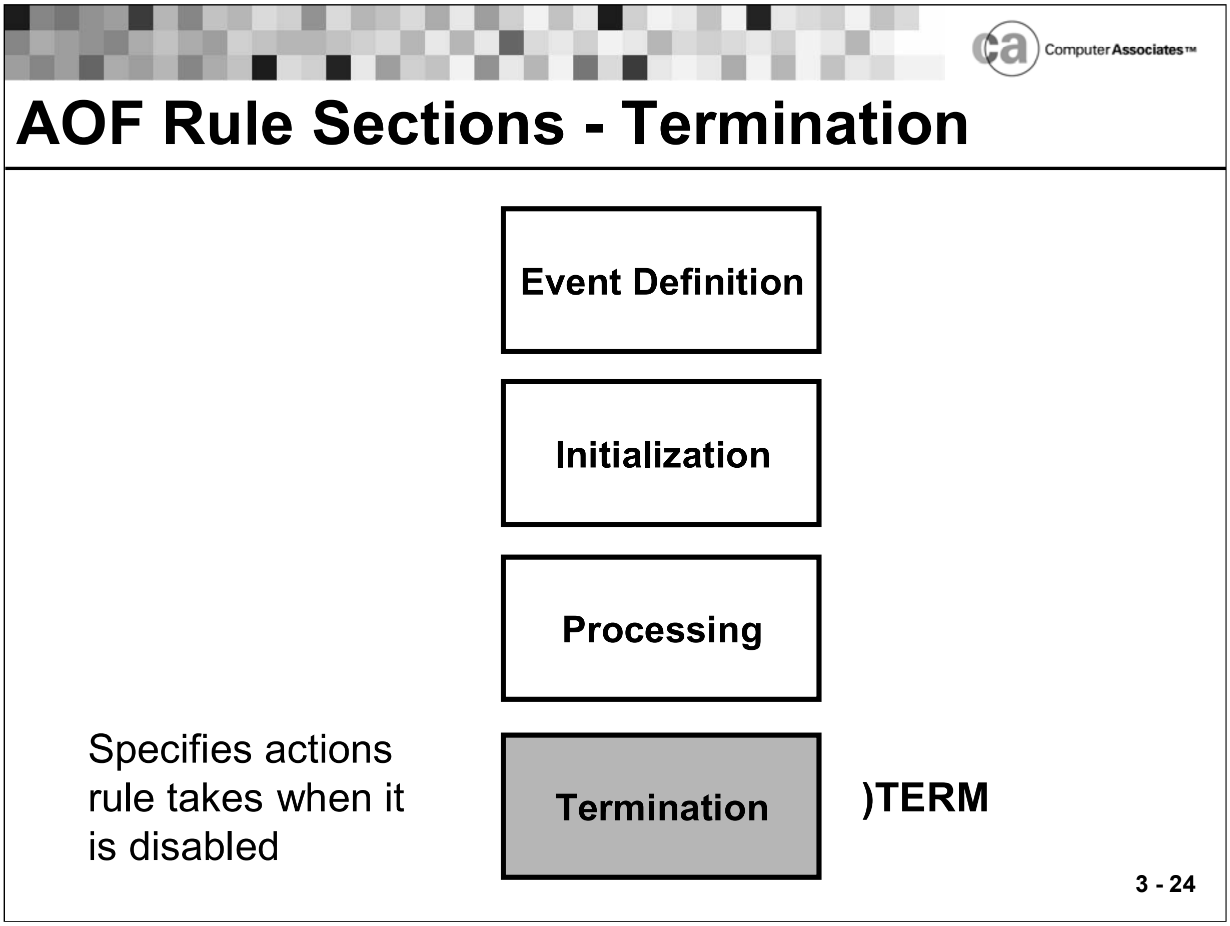
Processing Example

```
)MSG IEF450I
)INIT
/*This code will fire ONCE when the rule is enabled*/
IF OPSINFO('SMFID') <> 'SYSA' THEN RETURN 'REJECT'
TABENDS = 0
)PROC
/* This code will fire each time AOF detects */
/* an IEF450I message event on the system.   */
TABENDS = TABENDS + 1
IF MSG.JOBNAME <> 'PRDCICSA' THEN RETURN
PARSE VAR MSG.TEXT . 'ABEND=' ABEND
CONSOLE = OPSINFO('MSTCONSNM')
ADDRESS WTO
   "MSGID(OPSAUTO1) TEXT('PRDCICSA ABEND CODE=",
   "("ABEND" AT "TIME()"') HILITE CNNAME("CONSOLE")"
```

3 - 23

Processing Example

The example builds upon the event definition and initialization examples. A combination of AOF tools (REXX, AOF variables, OPS/REXX host environments) is used to send an alert message to the sysplex master console if the job named PRDCICSA abends.



AOF Rule Sections - Termination

Specifies actions rule takes when it is disabled

3 - 24

Termination

The termination section of a rule specifies the action(s) the rule takes when it is disabled. The termination section is optional. If included, it is the last section of a rule. This section executes in the Unicenter CA-OPS/MVS main address space.

Use this format for coding the termination section:

```
)TERM
/* Insert termination section action(s) */
```

You can use the termination section to:

- Issue z/OS, JES3, or VM operator commands.
- Invoke TSO commands or CLISTs.
- Reset or save variable values.

Valid Return Values

- The OPS/REXX RETURN statement allows or disallows a rule’s disablement.
- Valid values for a RETURN statement in a rule’s termination section are:

  ACCEPT

  Allows the rule to be disabled.

  REJECT

  Prevents the rule from being disabled.
- Example. This statement prevents the rule from being disabled:

  ```
  RETURN "REJECT"
  ```
- If you do not specify a return value, the default return value is ACCEPT.
- If a run-time error occurs, the return value is ACCEPT (assuming that the error occurs while the termination section is executing).

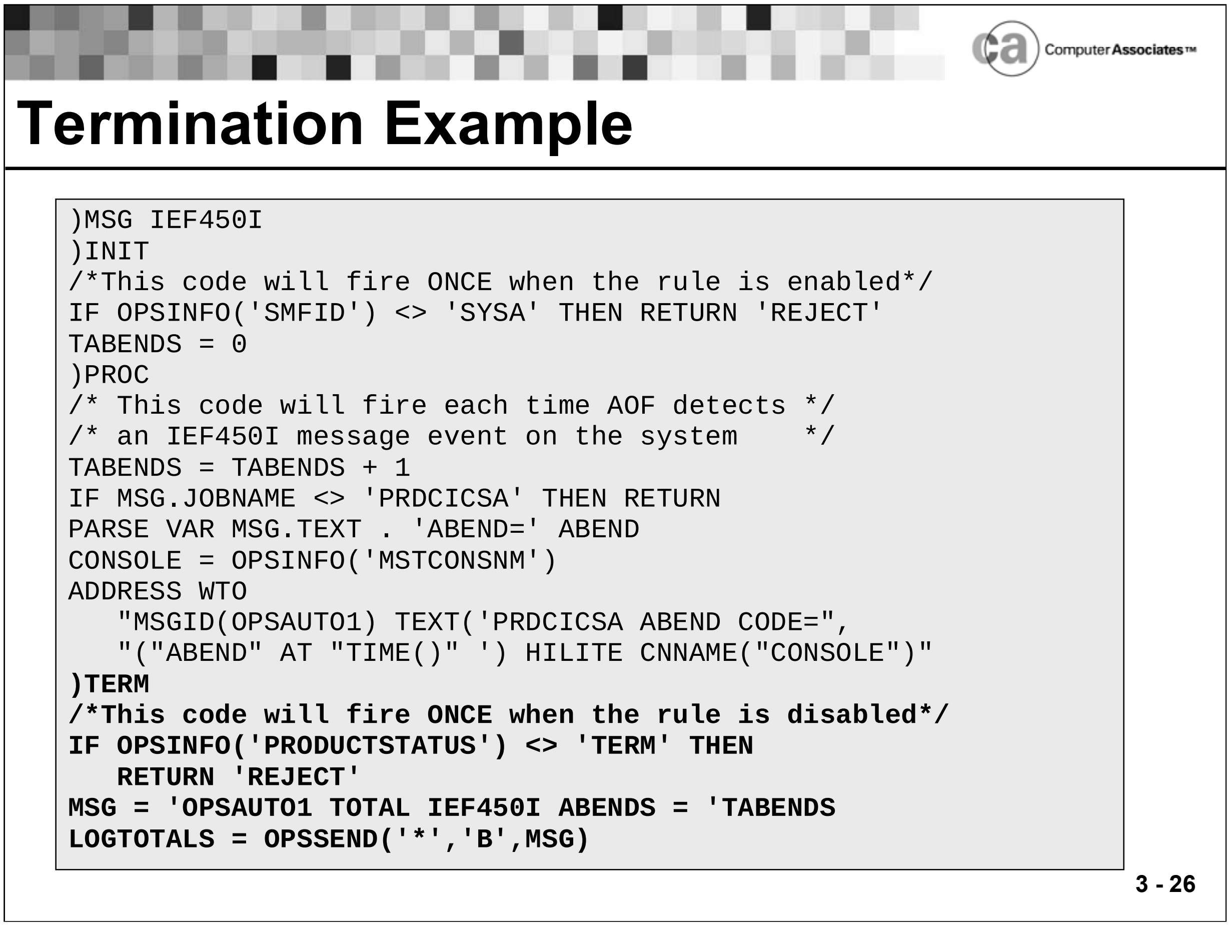
  Note: A return value of REJECT stops the disablement of a single rule only. If you disable the rule set, the rules within the rule set are always disabled, regardless of the return values in the individual rules.

Specifying End of a Rule

The END statement marks the end of a rule. The END statement is optional and does not affect rule execution. If included, it is always the last line of a rule.

Use this format for coding the END statement:

```
)END
```

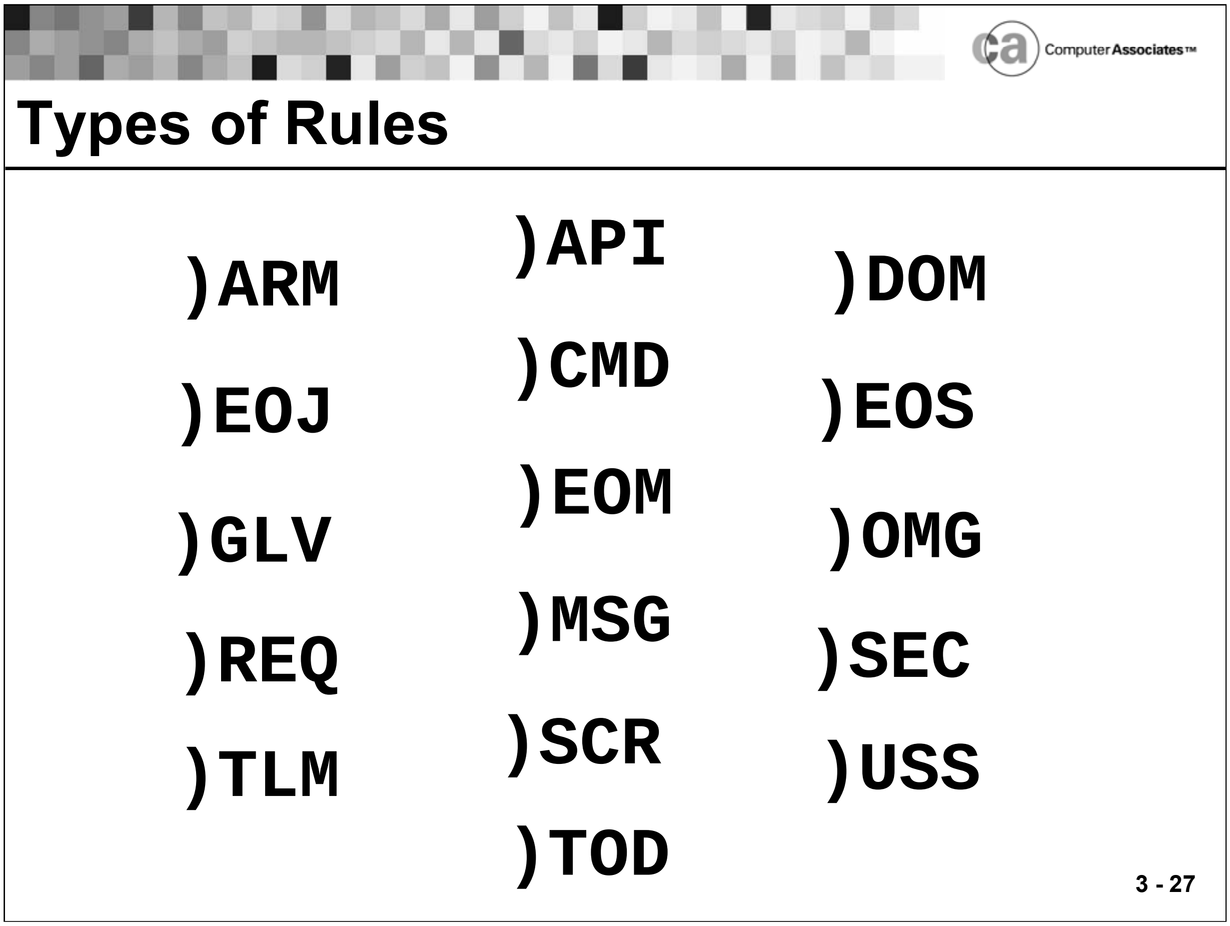


Termination Example

```
)MSG IEF450I
)INIT
/*This code will fire ONCE when the rule is enabled*/
IF OPSINFO('SMFID') <> 'SYSA' THEN RETURN 'REJECT'
TABENDS = 0
)PROC
/* This code will fire each time AOF detects */
/* an IEF450I message event on the system    */
TABENDS = TABENDS + 1
IF MSG.JOBNAME <> 'PRDCICSA' THEN RETURN
PARSE VAR MSG.TEXT . 'ABEND=' ABEND
CONSOLE = OPSINFO('MSTCONSNM')
ADDRESS WTO
   "MSGID(OPSAUTO1) TEXT('PRDCICSA ABEND CODE=",
   "("ABEND" AT "TIME()" ') HILITE CNNAME("CONSOLE")"
)TERM
/*This code will fire ONCE when the rule is disabled*/
IF OPSINFO('PRODUCTSTATUS') <> 'TERM' THEN
   RETURN 'REJECT'
MSG = 'OPSAUTO1 TOTAL IEF450I ABENDS = 'TABENDS
LOGTOTALS = OPSSEND('*','B',MSG)
```

Termination Example

The example, which builds upon the event definition, initialization, and processing examples. Rule disablement is allowed only at Unicenter CA-OPS/MVS shutdown. The example code also sends a message to the OPSLOG with the value of the static variable TABENDS, which was calculated during the rule's processing section.



Types of Rules

)ARM

)EOJ

)GLV

)REQ

)TLM

)API

)CMD

)EOM

)MSG

)SCR

)TOD

)DOM

)EOS

)OMG

)SEC

)USS

3 - 27

Types of Rules

There are 15 rule types in Unicenter CA-OPS/MVS:

- ARM - Responds to an Automatic Restart Management (ARM) event, providing the ability to intercept the event before the restart occurs.
- API - Responds to direct interface calls from other CA products.
- CMD - Responds to a command event.
- DOM - Responds to a delete-operator-message event.
- EOJ - Responds to an end-of-job event.
- EOM - Responds to an end-of-memory event.
- EOS - Responds to an end-of-step event, providing ability to monitor termination of each step of initiated batch jobs, and started tasks.

- GLV

  Responds to a global variable event, allowing you to create an inferencing capability within Unicenter CA-OPS/MVS. Inferencing refers to one event triggering another, which triggers another, and so on, until such processing achieves some goal.

- MSG

  Responds to a message event.

- OMG

  Responds to an OMEGAMON exception event.

- REQ

  Responds to an end-user request event.

- SCR

  Responds to a screen event.

- SEC

  Responds to a security event, providing an easy-to-use method for protecting Unicenter CA-OPS/MVS facilities.

- TLM

  Responds to a time limit excession event, providing the ability to intercept the event when either processor usage or continuous wait time limits for a job or task are exceeded.

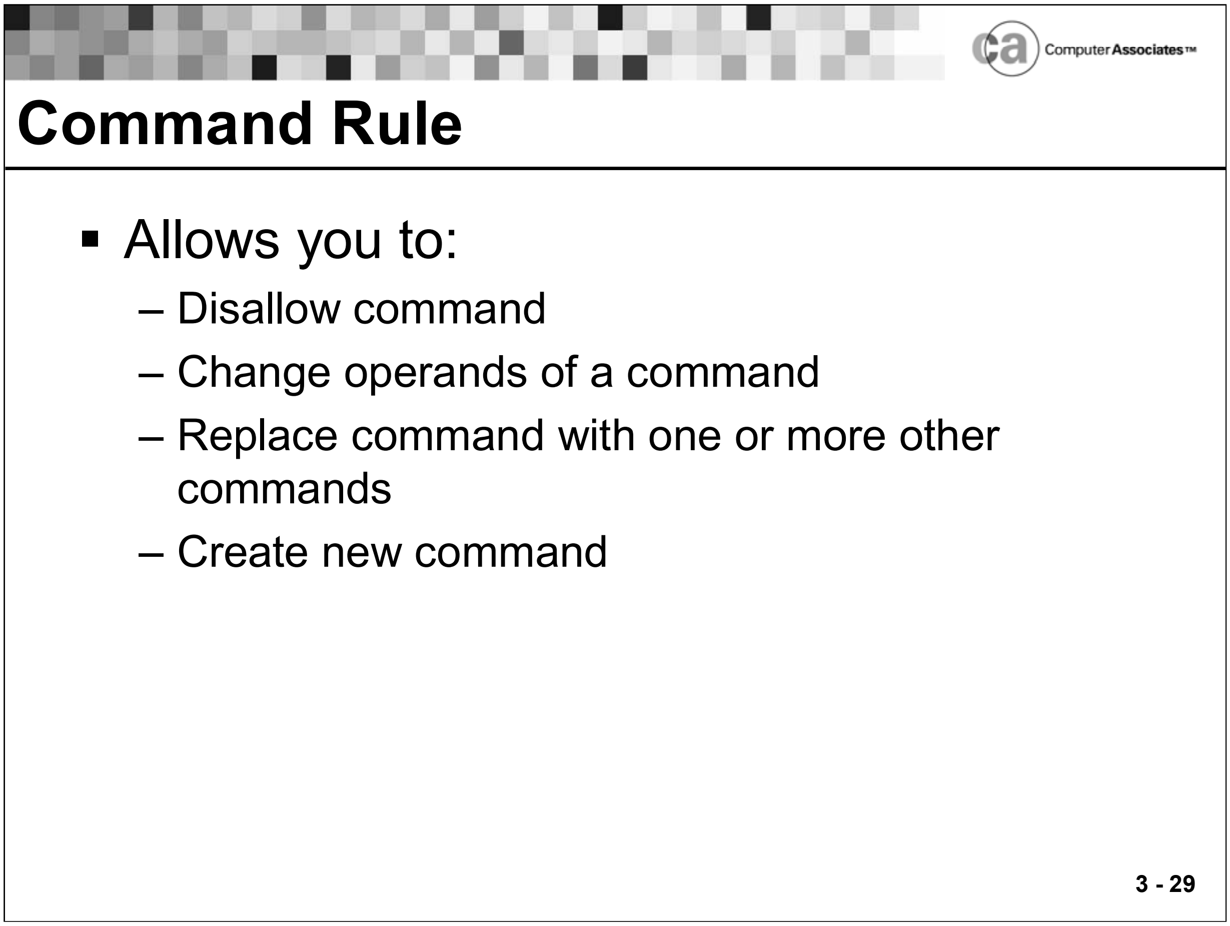
- TOD

  Responds to a time-of-day event, providing the ability to take some action at a certain time or after a specified time interval.

- USS

  Responds to a UNIX System Services event, allowing you to write automation procedures for messages that originate from Unicenter TNG Framework for z/OS.

Command, message, security, and time-of-day rules are covered in detail in this lesson.



Command Rule

- Allows you to:
  - Disallow command
  - Change operands of a command
  - Replace command with one or more other commands
  - Create new command

3 - 29

Command Rule

A rule responds to a command if the first blank-delimited word in the command string matches the command verb string that you specify in the rule's event definition section.

Use this format when coding the command event definition section:

```
)CMD cmdverbspec
```

Follow these guidelines when specifying the character string for the cmdverbspec command verb specifier:

- Specify one to ten characters.
- The string cannot contain embedded blank spaces.
- You can use the wildcard (*) character.

  ST* matches z/OS START and STOP commands or any pseudo command that begins with ST.

  * alone matches all command events on the system.

- Lowercase letters are acceptable, but the AOF converts them to uppercase for event testing.
- z/OS and VM command considerations

  Specify the full command verb or a mask of the command verb when attempting to trigger z/OS and VM commands. For example, when writing a command rule that triggers whenever a z/OS DISPLAY command is issued, specify)CMD DISPLAY rather than)CMD D, or specify)CMD MODIFY rather than)CMD F to execute z/OS modify commands. Although you must specify the full z/OS or VM command verb in the rule's event definition section, a command rule recognizes a command event if an operator issues a short form (alias) of the command. Additional logic can be implemented in the)PROC section of the rule to interrogate the CMD.TEXT event variable to see the exact z/OS or VM command that was entered in the SSI.

  z/OS may reissue some commands internally if they do not originate from the CONSOLE address space (that is, if a program issues the commands rather than a z/OS console). z/OS re-issues such commands so that the processing occurs in the CONSOLE address space, causing a CMD rule to possibly execute twice. z/OS reissues DISPLAY ACTIVE commands and any other command that creates paged-frame display output on a z/OS console.
- JES2 command considerations

  Specify the JES2 command character followed by the first letter of the JES2 command when attempting to trigger JES2 commands. For example, assuming the "$" sign is the JES2 command character and you want to trigger the $TI initiator command, you would code a command specifier of)CMD $T. Additional logic can be implemented in the)PROC section of the rule to interrogate the CMD.TEXT event variable to view the exact JES2 command that was entered.
- JES3 command considerations

  When writing rules that respond to JES3 commands, begin the cmdverbspec event identifier string with the wildcard (*) character; for example,)CMD *START.

Because the cmdverbspec string in the event definition section must begin with the wildcard (*) character, a rule triggered by an INQUIRY command is also triggered by NOINQUIRY (not a JES3 command) and INQUIRY (a possible z/OS command).You can solve the problem by coding the processing section of your rule as shown in this example:

```
)CMD *INQUIRY
)PROC
  IF CMD.VERB ¬= '*INQUIRY' THEN
    RETURN 'NOACTION'
```

If a JES3 command originates from an MCS console, the AOF processes the rule twice (once as a z/OS command and once as a JES3 command), meaning that a rule triggered by the command executes twice.

A CMD rule can change *START PRTR1 to *START PRTR2, but it cannot change it to VARY 2F0,OFFLINE.

- Subsystem command character considerations other than JES2 commands

  Specify the command character followed by the wildcard * character to execute commands issued via a defined command character. For example, assuming the “/” sign is the command character for a particular subsystem, you would code)CMD /* to trigger commands issued via the command character. Additional logic can be implemented in the)PROC section of the rule to interrogate the CMD.TEXT event variable to view the exact command that was entered.

Valid Return Values

The OPS/REXX RETURN statement specifies the final disposition of a z/OS command. For example, the RETURN statement can:

- Allow z/OS to execute a command.
- Prevent z/OS from executing a command.
- Force z/OS to reject a command.

Valid values for a RETURN statement in the processing section of a command rule are:

- NOACTION

  Allows z/OS to process a command (after AOF processing, if any).

- ACCEPT

  Prevents z/OS from processing a command. Instead, the AOF allows your rule to process the command.

- REJECT

  Causes z/OS to reject a command as invalid (regardless of whether the command is a valid or invalid z/OS command). Use this option to prevent a z/OS command from executing.

- Default —RETURN "NOACTION"
- Notes: The return values listed here are character constants, rather than keywords.
- An unrecognized return value—for example, a misspelled value—defaults to a value of NOACTION.

Other RETURN Statement Considerations

In a command rule, the return value can affect command processing as follows:

- If multiple rules respond to a single command event, the AOF uses the highest-precedence return value. The order of precedence is:

  REJECT (highest precedence)

  ACCEPT

  NOACTION (lowest precedence)

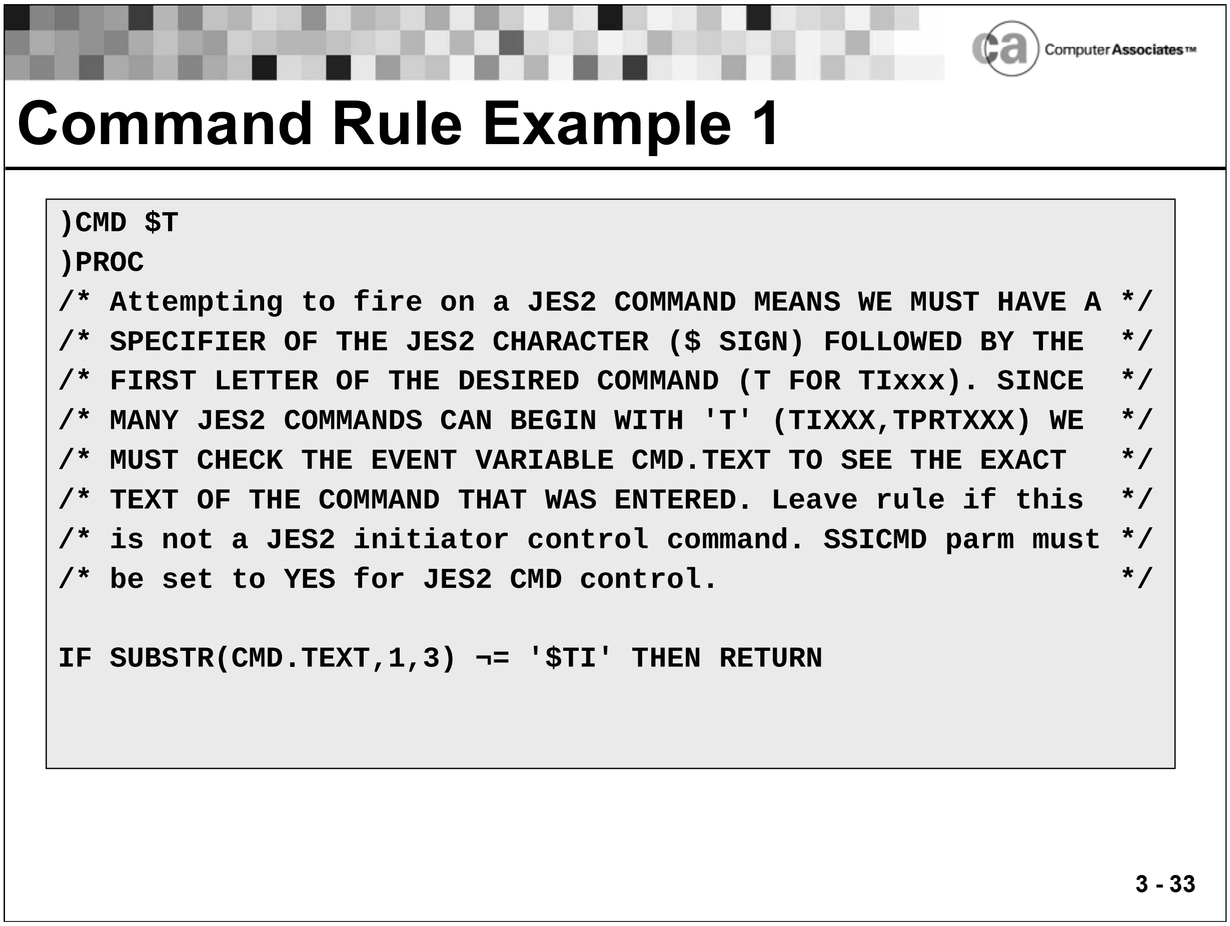
- The return value does not necessarily affect how subsystems process commands.

  Some types of commands (such as z/OS and VTAM) are processed only after all other subsystems (including Unicenter CA-OPS/MVS) have processed them. You can control such commands by either:

  Setting one of the three valid return values

  Using the rule to change the command text (modifying the text of subsystem commands such as JES, DB2, NetView, BDT, and so on may not work; see your Unicenter CA-OPS/MVS documentation for more information).

  Note: If the SSICMD (Unicenter CA-OPS/MVS initialization) parameter is set to YES, a REJECT return value assures that the AOF intercepts a command before any other subsystem receives it.



Command Rule Example 1

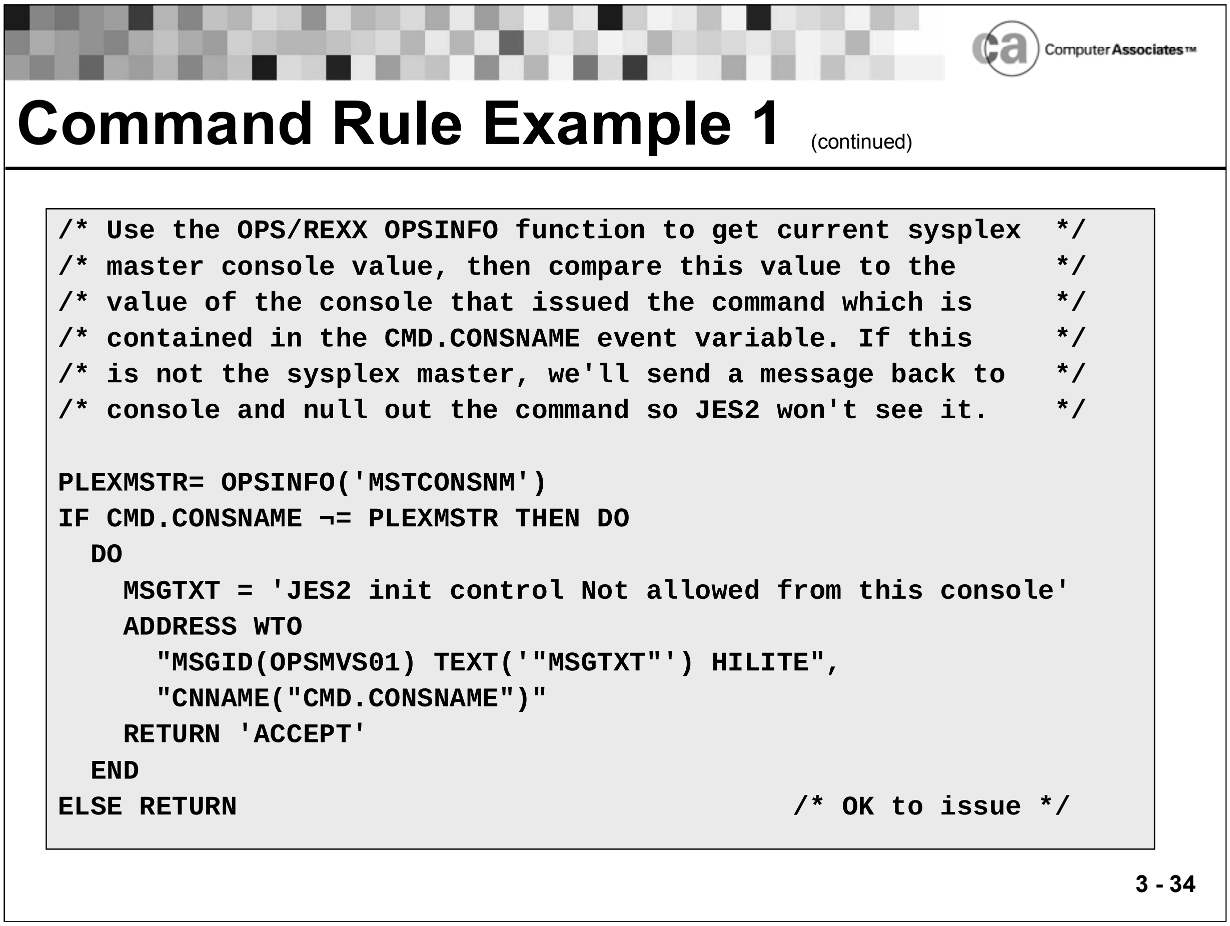
```
)CMD $T
)PROC
/* Attempting to fire on a JES2 COMMAND MEANS WE MUST HAVE A */
/* SPECIFIER OF THE JES2 CHARACTER ($ SIGN) FOLLOWED BY THE  */
/* FIRST LETTER OF THE DESIRED COMMAND (T FOR TIxxx). SINCE  */
/* MANY JES2 COMMANDS CAN BEGIN WITH 'T' (TIXXX,TPRTXXX) WE  */
/* MUST CHECK THE EVENT VARIABLE CMD.TEXT TO SEE THE EXACT   */
/* TEXT OF THE COMMAND THAT WAS ENTERED. Leave rule if this  */
/* is not a JES2 initiator control command. SSICMD parm must */
/* be set to YES for JES2 CMD control.                       */

IF SUBSTR(CMD.TEXT,1,3) ¬= '$TI' THEN RETURN
```

3 - 33

Command Rule Example 1

The above example is a rule that ensures that JES2 initiator control commands ($Tix) can only be issued from the current sysplex master console.

First, the rule checks to see if an initiator control command caused the rule to fire. If not, the rule ends without taking any action.



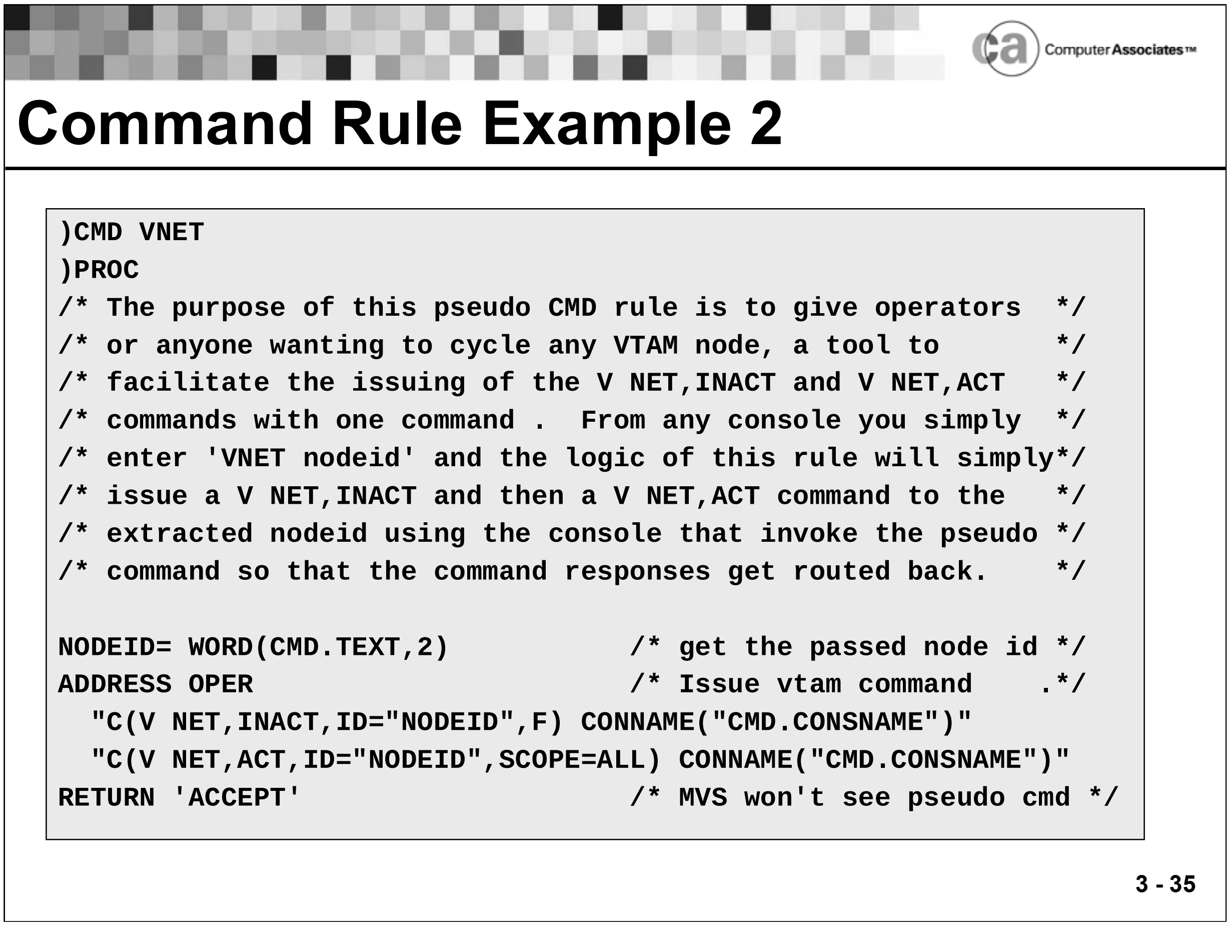
Command Rule Example 1 (continued)

```
/* Use the OPS/REXX OPSINFO function to get current sysplex  */
/* master console value, then compare this value to the      */
/* value of the console that issued the command which is     */
/* contained in the CMD.CONSNAME event variable. If this     */
/* is not the sysplex master, we'll send a message back to   */
/* console and null out the command so JES2 won't see it.    */

PLEXMSTR= OPSINFO('MSTCONSNM')
IF CMD.CONSNAME ¬= PLEXMSTR THEN DO
  DO
    MSGTXT = 'JES2 init control Not allowed from this console'
    ADDRESS WTO
      "MSGID(OPSMVS01) TEXT('"MSGTXT"') HILITE",
      "CNNAME("CMD.CONSNAME")"
    RETURN 'ACCEPT'
  END
ELSE RETURN                                  /* OK to issue */
```

3 - 34

Command Rule Example 1 (cont.)

If it was an initiator control command that caused the rule to fire, then the rule uses the OPSINFO function to retrieve the name of the master console. It then verifies that the command was issued from the master console. If not, the rule rejects the command and sends a WTO back to the originating console that indicates initiator control commands are only allowed at the master console.



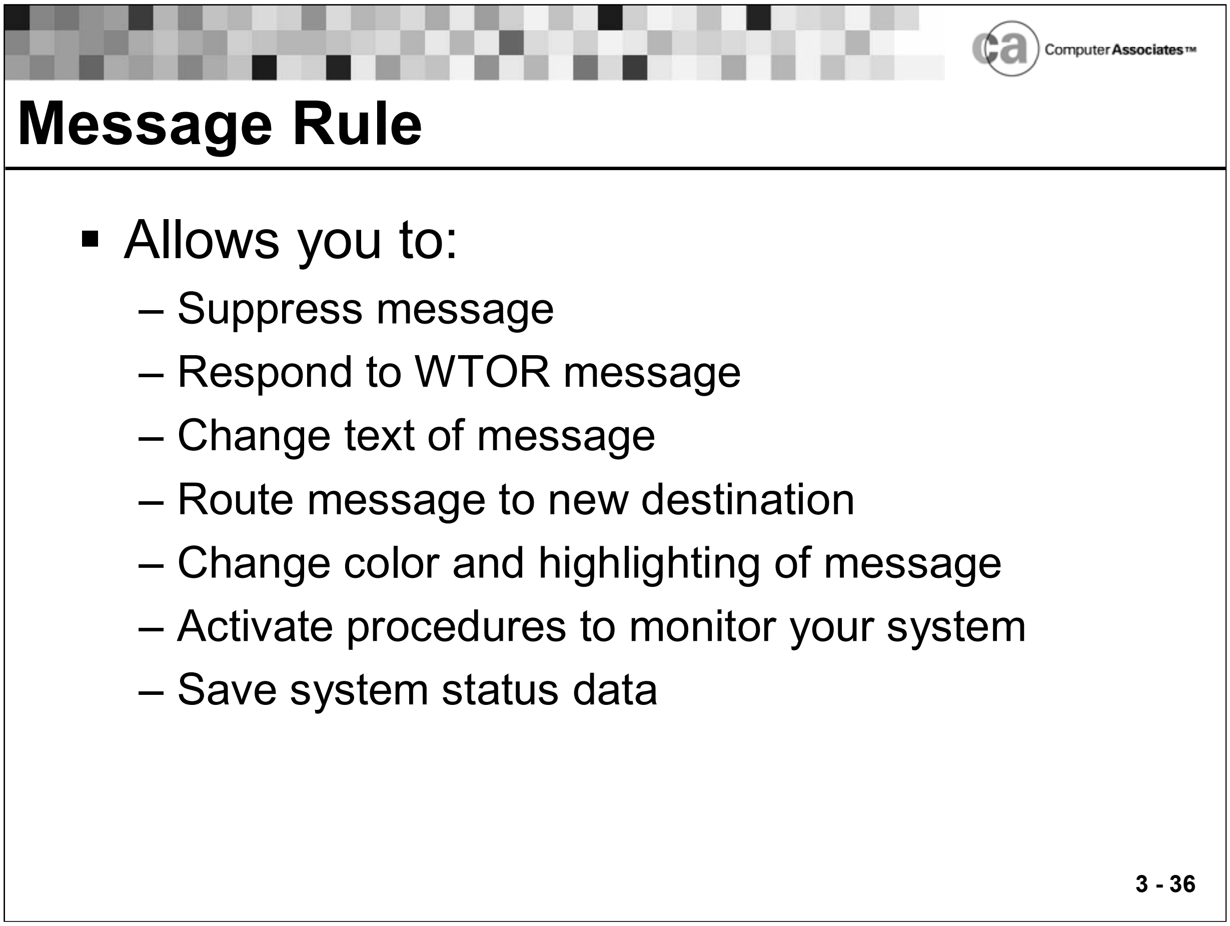
Command Rule Example 2

```
)CMD VNET
)PROC
/* The purpose of this pseudo CMD rule is to give operators  */
/* or anyone wanting to cycle any VTAM node, a tool to       */
/* facilitate the issuing of the V NET,INACT and V NET,ACT   */
/* commands with one command .  From any console you simply  */
/* enter 'VNET nodeid' and the logic of this rule will simply*/
/* issue a V NET,INACT and then a V NET,ACT command to the   */
/* extracted nodeid using the console that invoke the pseudo */
/* command so that the command responses get routed back.    */

NODEID= WORD(CMD.TEXT,2)           /* get the passed node id */
ADDRESS OPER                       /* Issue vtam command    .*/
  "C(V NET,INACT,ID="NODEID",F) CONNAME("CMD.CONSNAME")"
  "C(V NET,ACT,ID="NODEID",SCOPE=ALL) CONNAME("CMD.CONSNAME")"
RETURN 'ACCEPT'                    /* MVS won't see pseudo cmd */
```

3 - 35

Command Rule Example 2

The example is of a rule that uses a pseudo command rule to cycle a VTAM node.



Message Rule

- Allows you to:
  - Suppress message
  - Respond to WTOR message
  - Change text of message
  - Route message to new destination
  - Change color and highlighting of message
  - Activate procedures to monitor your system
  - Save system status data

3 - 36

Message Rule

A rule responds to a message if the message identifier matches the rule’s message ID specification. The message identifier is usually the first word of the message.

Use this format when coding the message event definition section:

```
)MSG msgidspec [NOOPSLOG]
```

Follow these guidelines when specifying the character string for the msgidspec message ID specifier:

- Specify one to ten characters.
- The string cannot contain embedded blank spaces.
- You can use the wildcard (*) character.
- Lowercase letters are acceptable, but the AOF converts them to uppercase for event testing.

- In general, Unicenter CA-OPS/MVS messages cannot be processed by the AOF. However, messages that have a severity code of O or J are exceptions to this rule.
- You can change the severity code of a Unicenter CA-OPS/MVS message ID by using the OPSPRM() REXX function in the OPSSPA00 member of the SYS1.PARMLIB data set.
- Example: The OPS2085O message is one example of a message that is automateable by default. Suppose that you want to change the severity code of message so that no message rule responds to it. You could use this statement:

```
T = OPSPRM_Set("OPS2085","H")
```

NOOPSLOG Option

The NOOPSLOG option prevents a message (specified by msgidspec) from appearing in the OPSLOG.

Warning! Use the NOOPSLOG option carefully to avoid losing system information necessary for effective automation.

Note: You cannot remove all message event records from the OPSLOG. For example, specifying MSG * NOOPSLOG causes the AOF to ignore the NOOPSLOG option. The NOOPSLOG option will be ignored if the msgidspec contains an imbedded wildcard (*) character (for example, IST*I). The NOOPSLOG option is acknowledged only for complete msgidspecs (for example, IST123I) or for a prefix msgidspec (for example, IST*).

Valid Return Values

The OPS/REXX RETURN statement specifies the final disposition of a message. The RETURN statement can:

- Allow z/OS to route a message normally.
- Prevent a message from appearing on the console.
- Prevent a message from appearing in the SYSLOG log file.
- Suppress a message.
- Override a return value specified by a prior rule.

Valid values for a RETURN statement in the processing section of a message rule are:

- NORMAL

  Allows z/OS to route a message normally.

- SUPPRESS

  Prevents a message from appearing on the console. The message appears in the OPSLOG.

  Note: IMS command response segment messages that are destined for the MTO terminal cannot be modified or canceled. For messages of this type, the SUPPRESS value is ignored.

- DISPLAY

  Prevents a message from appearing in the SYSLOG log file. The message appears on the console.

- DELETE

  Suppresses a message entirely. The message does not appear on the console or in the SYSLOG log file.

- FORCENORMAL

  Allows z/OS to route a message normally, but overrides any return value that was specified by a prior rule.

- Default —RETURN "NORMAL".
- Notes: The return values listed here are character constants, rather than keywords.
- An unrecognized return value—for example, a misspelled value—defaults to a value of NORMAL.

Other RETURN Statement Considerations

Consider the following when specifying the RETURN statement in the processing section of a message rule:

- If multiple rules respond to a single message event, the AOF uses the highest-precedence return value. The order of precedence is:

  DELETE (highest precedence)

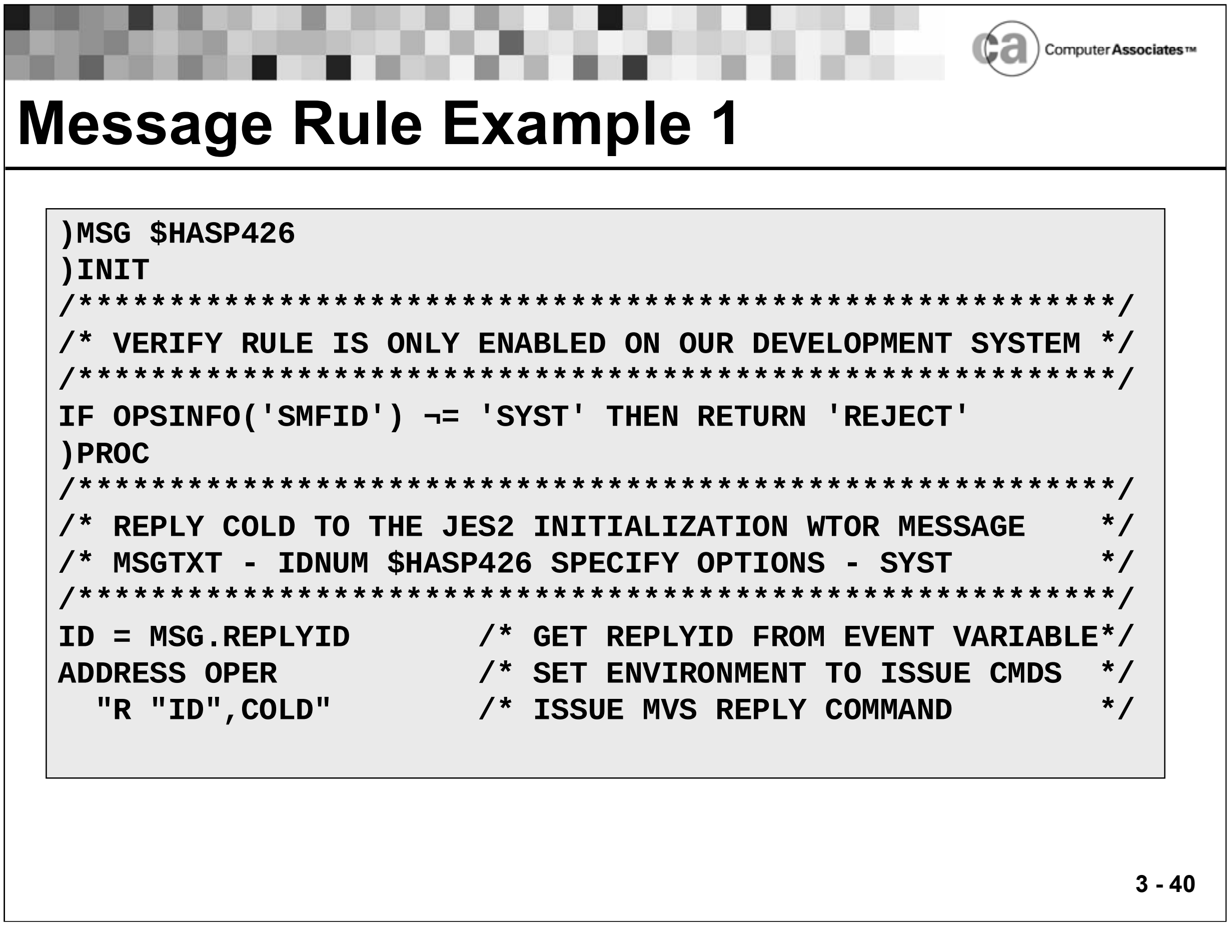
  DISPLAY

  SUPPRESS

  NORMAL (lowest precedence)

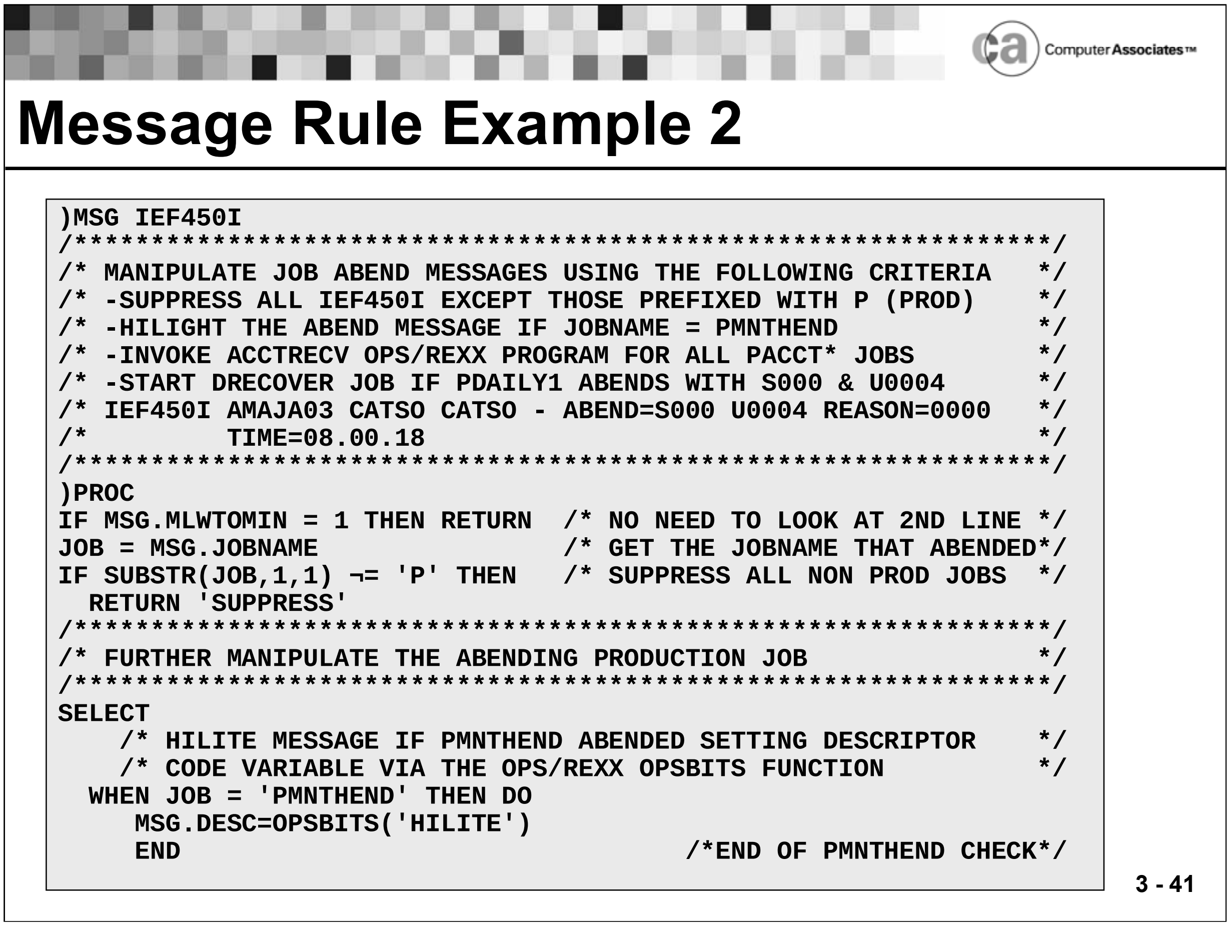
- Note: The FORCENORMAL return value overrides the order of precedence and is meaningful when multiple rules execute for a message. For example, if the first rule returns the SUPPRESS value and the last rule returns the FORCENORMAL value, the message appears on the appropriate consoles. However, subsequent rules may successfully use the SUPPRESS, DISPLAY, and DELETE return values.

The DELETE and DISPLAY return values work as described only if the AOFDELETE parameter is set to YES (the default setting). If the AOFDELETE parameter is set to NO, the rule processes a message as though the DELETE return value is SUPPRESS and the DISPLAY return value is NORMAL.



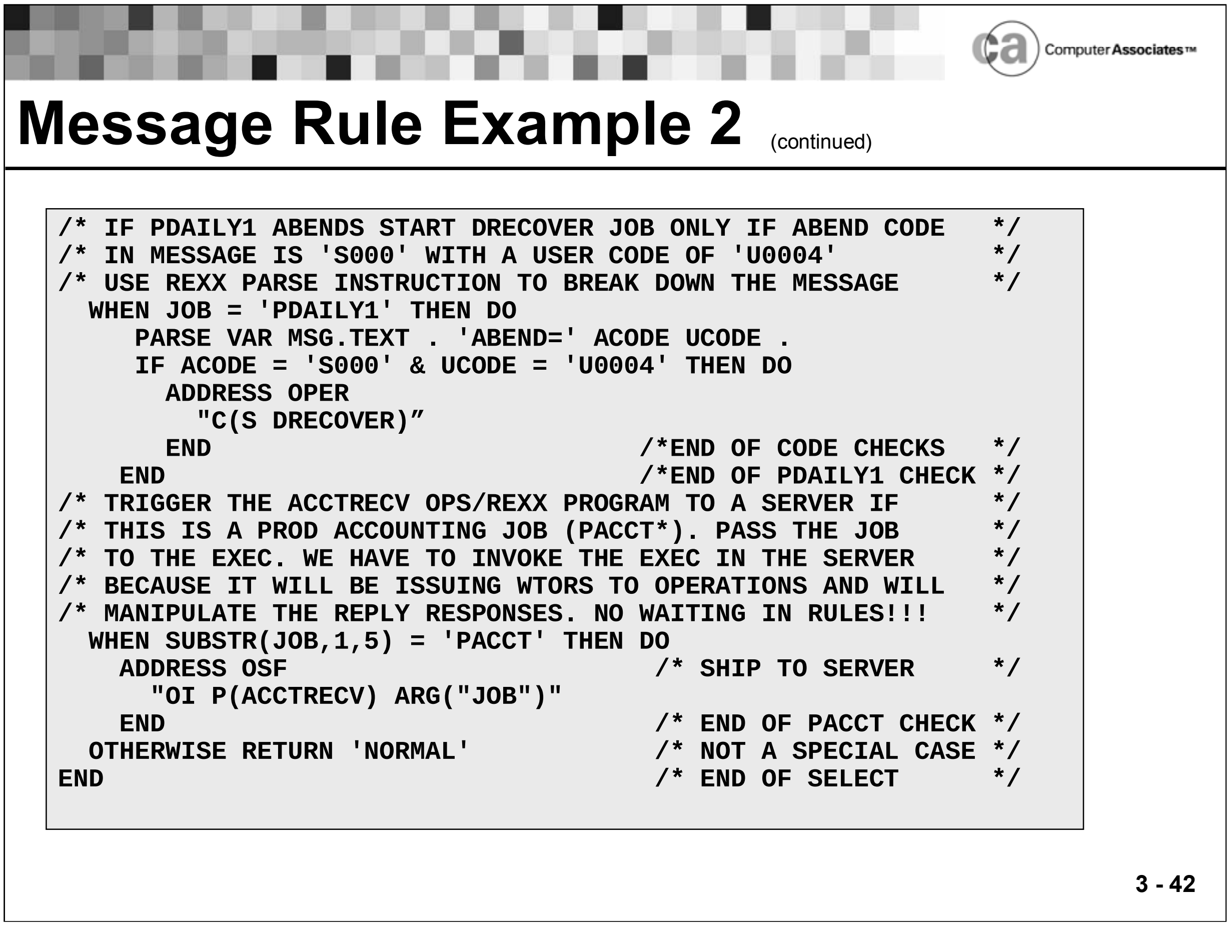
Message Rule Example 1

```
)MSG $HASP426
)INIT
/*********************************************************/
/* VERIFY RULE IS ONLY ENABLED ON OUR DEVELOPMENT SYSTEM */
/*********************************************************/
IF OPSINFO('SMFID') ¬= 'SYST' THEN RETURN 'REJECT'
)PROC
/*********************************************************/
/* REPLY COLD TO THE JES2 INITIALIZATION WTOR MESSAGE    */
/* MSGTXT - IDNUM $HASP426 SPECIFY OPTIONS - SYST        */
/*********************************************************/
ID = MSG.REPLYID       /* GET REPLYID FROM EVENT VARIABLE*/
ADDRESS OPER           /* SET ENVIRONMENT TO ISSUE CMDS  */
  "R "ID",COLD"        /* ISSUE MVS REPLY COMMAND        */
```



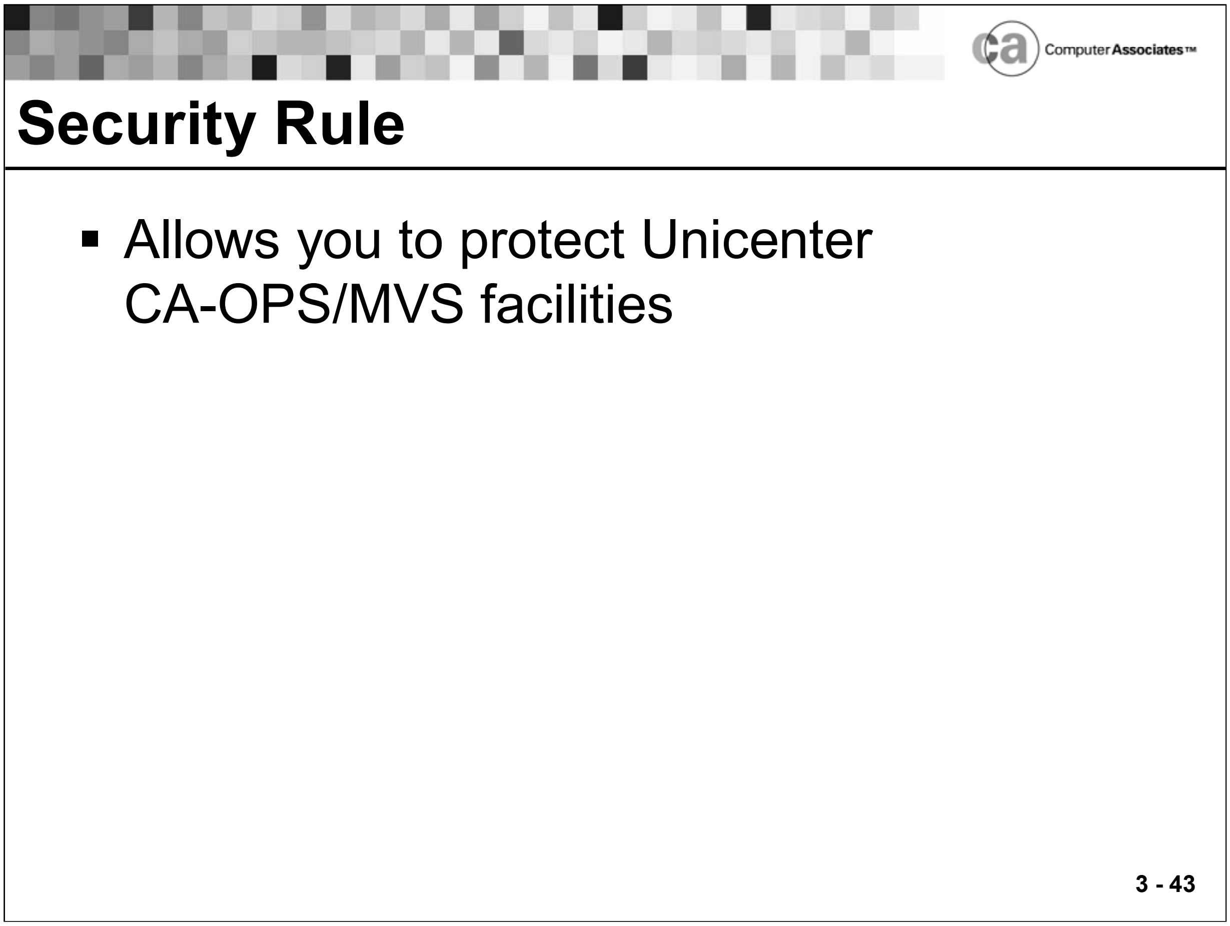
Message Rule Example 2

```
)MSG IEF450I
/****************************************************************/
/* MANIPULATE JOB ABEND MESSAGES USING THE FOLLOWING CRITERIA   */
/* -SUPPRESS ALL IEF450I EXCEPT THOSE PREFIXED WITH P (PROD)    */
/* -HILIGHT THE ABEND MESSAGE IF JOBNAME = PMNTHEND             */
/* -INVOKE ACCTRECV OPS/REXX PROGRAM FOR ALL PACCT* JOBS        */
/* -START DRECOVER JOB IF PDAILY1 ABENDS WITH S000 & U0004      */
/* IEF450I AMAJA03 CATSO CATSO - ABEND=S000 U0004 REASON=0000   */
/*         TIME=08.00.18                                        */
/****************************************************************/
)PROC
IF MSG.MLWTOMIN = 1 THEN RETURN  /* NO NEED TO LOOK AT 2ND LINE */
JOB = MSG.JOBNAME                /* GET THE JOBNAME THAT ABENDED*/
IF SUBSTR(JOB,1,1) ¬= 'P' THEN   /* SUPPRESS ALL NON PROD JOBS  */
  RETURN 'SUPPRESS'
/****************************************************************/
/* FURTHER MANIPULATE THE ABENDING PRODUCTION JOB               */
/****************************************************************/
SELECT
    /* HILITE MESSAGE IF PMNTHEND ABENDED SETTING DESCRIPTOR    */
    /* CODE VARIABLE VIA THE OPS/REXX OPSBITS FUNCTION          */
  WHEN JOB = 'PMNTHEND' THEN DO
     MSG.DESC=OPSBITS('HILITE')
     END                                 /*END OF PMNTHEND CHECK*/
```



Message Rule Example 2 (continued)

```
/* IF PDAILY1 ABENDS START DRECOVER JOB ONLY IF ABEND CODE   */
/* IN MESSAGE IS 'S000' WITH A USER CODE OF 'U0004'          */
/* USE REXX PARSE INSTRUCTION TO BREAK DOWN THE MESSAGE      */
  WHEN JOB = 'PDAILY1' THEN DO
     PARSE VAR MSG.TEXT . 'ABEND=' ACODE UCODE .
     IF ACODE = 'S000' & UCODE = 'U0004' THEN DO
       ADDRESS OPER
         "C(S DRECOVER)”
       END                            /*END OF CODE CHECKS   */
    END                               /*END OF PDAILY1 CHECK */
/* TRIGGER THE ACCTRECV OPS/REXX PROGRAM TO A SERVER IF      */
/* THIS IS A PROD ACCOUNTING JOB (PACCT*). PASS THE JOB      */
/* TO THE EXEC. WE HAVE TO INVOKE THE EXEC IN THE SERVER     */
/* BECAUSE IT WILL BE ISSUING WTORS TO OPERATIONS AND WILL   */
/* MANIPULATE THE REPLY RESPONSES. NO WAITING IN RULES!!!    */
  WHEN SUBSTR(JOB,1,5) = 'PACCT' THEN DO
    ADDRESS OSF                        /* SHIP TO SERVER     */
      "OI P(ACCTRECV) ARG("JOB")"
    END                                /* END OF PACCT CHECK */
  OTHERWISE RETURN 'NORMAL'            /* NOT A SPECIAL CASE */
END                                    /* END OF SELECT      */
```



Security Rule

- Allows you to protect Unicenter CA-OPS/MVS facilities

3 - 43

Security Rule

A rule that responds to a security event provides an easy-to-use method for protecting Unicenter CA-OPS/MVS facilities. Unlike an assembler language authorization exit, a security rule is easy to write, implement, and update.

A security event occurs when a Unicenter CA-OPS/MVS facility is used.

Use this format when coding the security event definition section:

```
)SEC facility||eventqualifier
```

The facility security event specifier is one of the following character strings specifying a Unicenter CA-OPS/MVS facility:

- OPSAOF

  An ADDRESS AOF host command issued from within an OPS/REXX program.

- OPSBRW

  OPSBRW command processor used to view entries in the OPSLOG Browse facility.

- OPSCMD

  OPSCMD command processor or OPS/REXX ADDRESS OPER command, used to issue operator commands.

- OPSCTL

  ADDRESS OPSCTL host environment, used to control the Multi-System Facility (MSF).

- OPSDOM

  OPSDOM command processor used to delete an outstanding message.

- OPSEPI

  ADDRESS EPI host command issued from within an OPS/REXX program.

- OPSGLOBAL

  OPS/REXX global variable that is accessed or updated.

- OPSHFI

  OPSHFI command or REXX function, used to read, write, or delete variable records from shared VSAM file supporting global variables.

- OPSLOG

  Unicenter CA-OPS/MVS API request (processed by the Automation Analyzer).

- OPSOSF

  ADDRESS OSF host command issued from within an OPS/REXX program.

- OPSPARM

  OPSPARM command processor or OPS/REXX OPSPRM function used to change OPS/MVS parameter values

- OPSREPLY

  OPSREPLY command processor used to reply to WTOR messages.

- OPSREQ

  OPSREQ command processor, used to invoke AOF request rules.

- OPSRMT

  OPSRMT command processor used to issue a command to a remote system.

- OPSSMTBL

  OPSSMTBL command processor, used to maintain the directory table that System State Manager uses to manage tables containing system resource information.

- OPSWTO

  OPSWTO command processor or the ADDRESS WTO host environment, used to send WTO or WTOR messages.

- SQL

  OPSQL command processor or the ADDRESS SQL host environment, used to issue Structured Query Language (SQL) commands.

- SUBSYSDSN

  Unicenter CA-OPS/MVS subsystem data set that is opened.

- OPSVIEW

  OPSVIEW command processor used to invoke OPSVIEW interface

The eventqualifier value is a character string that specifies a subset of the facility security event specifier. Follow these guidelines when specifying the eventqualifier value:

- Concatenate the string with the facility string.
- Both strings must be connected with no blank spaces between them.
- For all facility values except OPSCMD and OPSGLOBAL, specify only a wildcard (*) character.
- Example —)SEC OPSBRW*

  For OPSCMD, specify a full command verb (rather than an abbreviation or alias), followed by the wildcard (*) character.

  For OPSGLOBAL, specify up to 41 characters of the global variable name, followed by the wildcard (*) character.

  Example: The facility and eventqualifier combination in this security event definition matches all global variables beginning with the GLOBAL1.FOOVAR prefix:

  ```
  )SEC OPSGLOBALGLOBAL1.FOOVAR*
  ```

- Lowercase letters are acceptable, but the AOF converts them to uppercase for event testing.

Valid Return Values

The OPS/REXX RETURN statement specifies the final disposition of a security event. The RETURN statement can:

- Refer a security event to the OPUSEX security exit.
- Force OPS/MVS to deny access to a requested facility.
- Allow access to a requested facility.
- Valid values for a RETURN statement in the processing section of a security rule are:

  NOACTION

  Allows the event to occur with no intervention from the AOF. Unicenter CA-OPS/MVS passes the event to the OPUSEX security exit after AOF processing (if any).

  ACCEPT

  Allows access to the requested facility and does not call the OPUSEX exit

  REJECT

  Denies access to the requested Unicenter CA-OPS/MVS facility and does not call the OPUSEX security exit

  Default —RETURN "NOACTION"

  Note: The return values listed here are character constants, rather than keywords.

  An unrecognized value—for example, a misspelled value—defaults to a value of NOACTION.

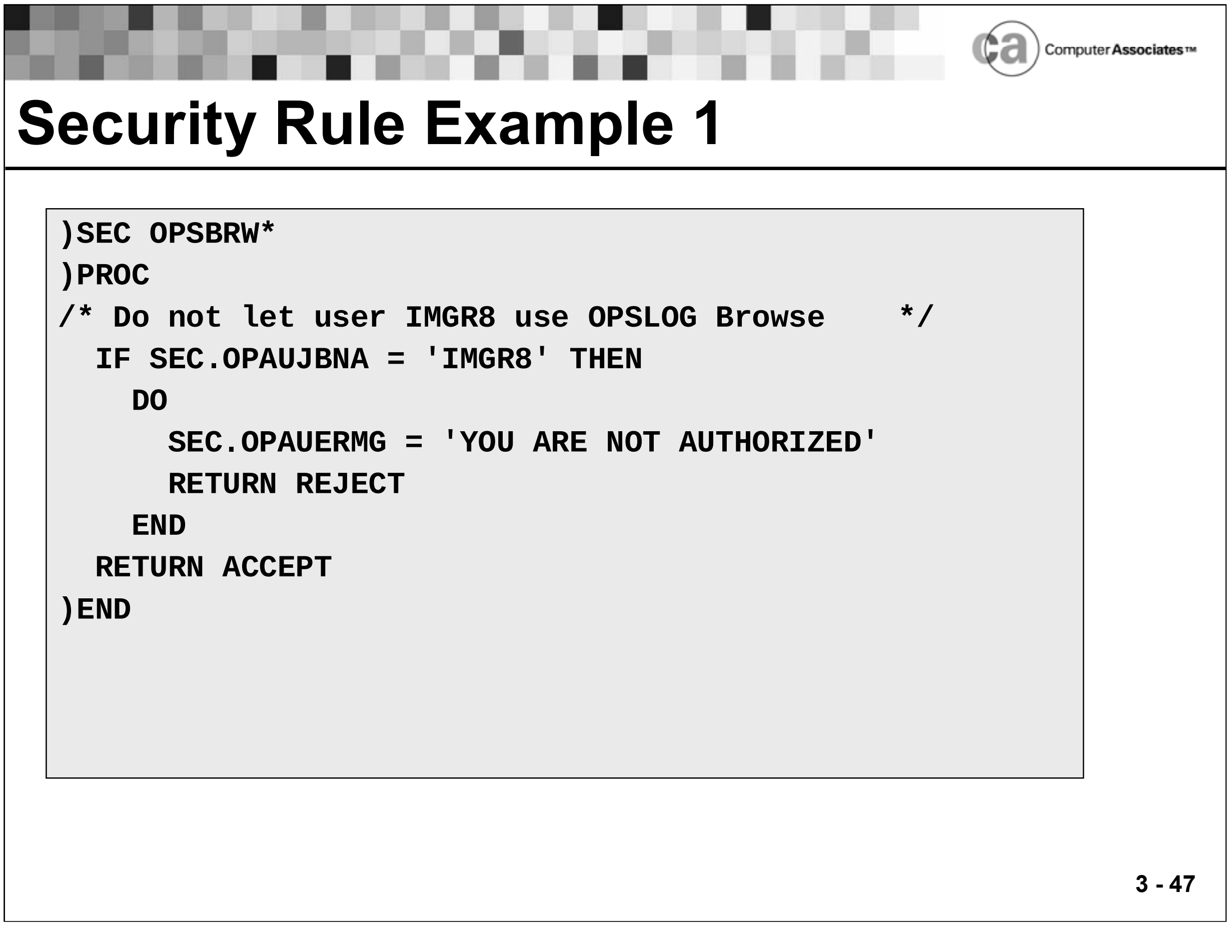
Other RETURN statement considerations:

- If multiple rules respond to a single security event, the AOF uses the highest-precedence return value. The order of precedence is:

  REJECT (highest)

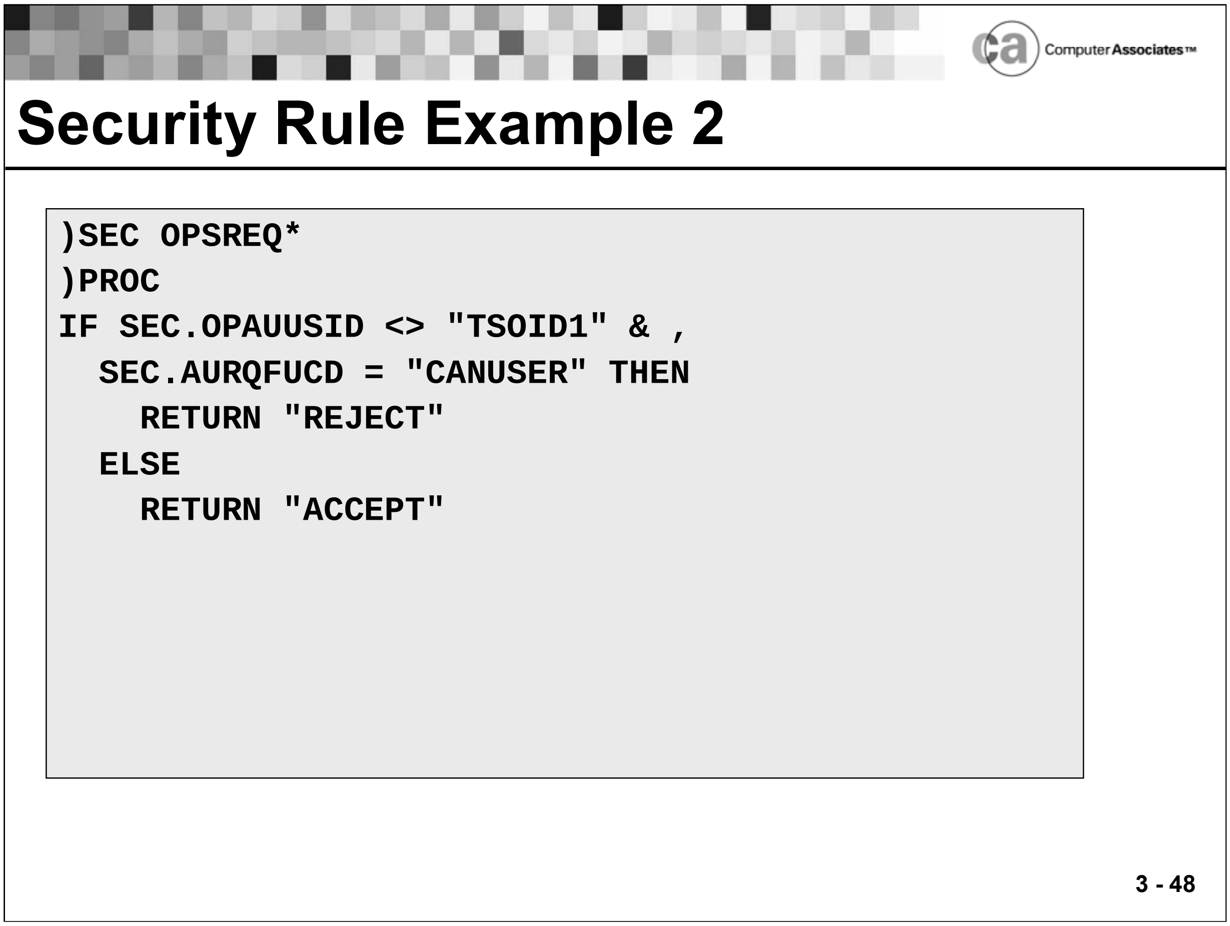
  ACCEPT

  NOACTION (lowest)



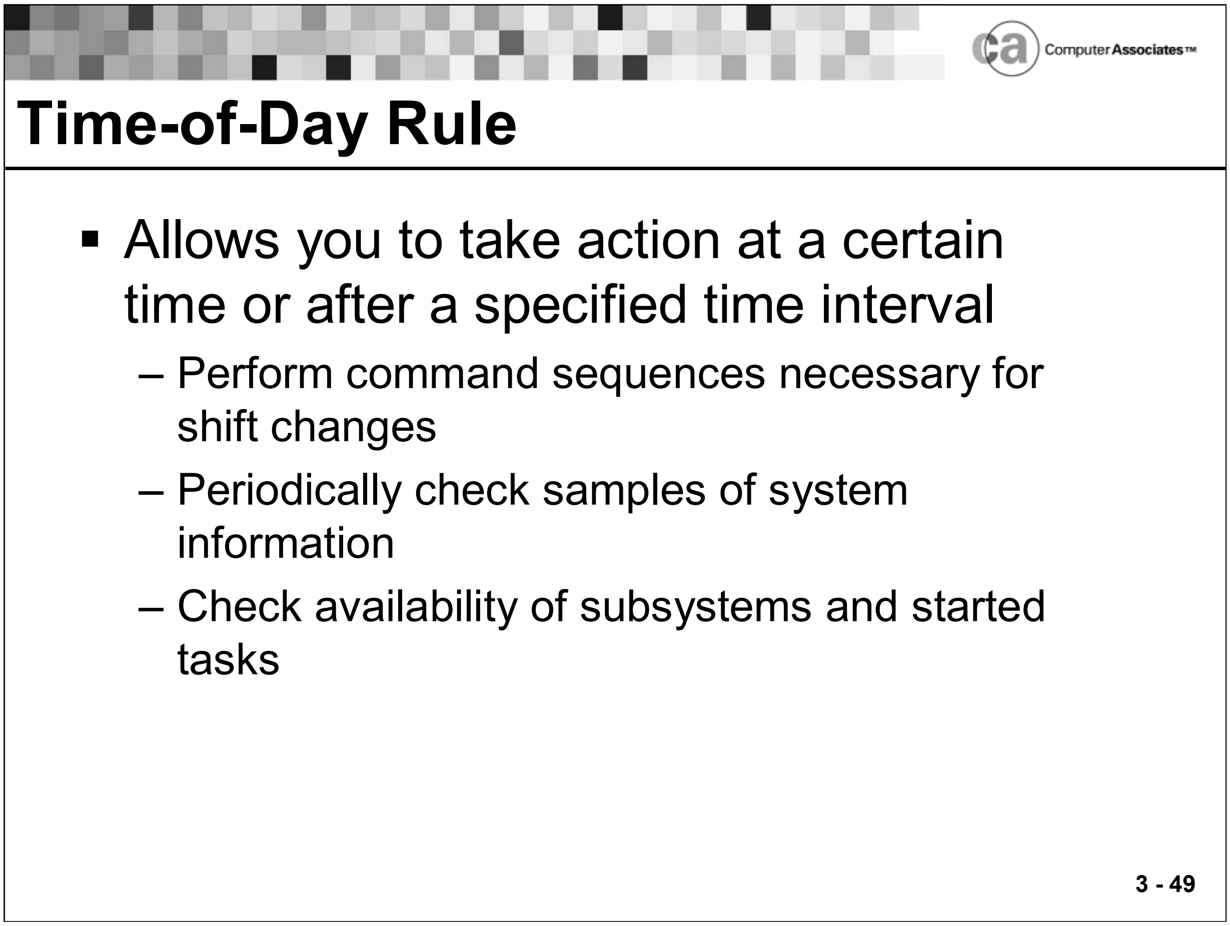
Security Rule Example 1

```
)SEC OPSBRW*
)PROC
/* Do not let user IMGR8 use OPSLOG Browse    */
  IF SEC.OPAUJBNA = 'IMGR8' THEN
    DO
      SEC.OPAUERMG = 'YOU ARE NOT AUTHORIZED'
      RETURN REJECT
    END
  RETURN ACCEPT
)END
```



Security Rule Example 2

```
)SEC OPSREQ*
)PROC
IF SEC.OPAUUSID <> "TSOID1" & ,
  SEC.AURQFUCD = "CANUSER" THEN
    RETURN "REJECT"
  ELSE
    RETURN "ACCEPT"
```



Time-of-Day Rule

- Allows you to take action at a certain time or after a specified time interval
  - – Perform command sequences necessary for shift changes
  - – Periodically check samples of system information
  - – Check availability of subsystems and started tasks

3 - 49

Time-of-Day Rule

A rule that responds to a time-of-day event takes some action at a certain time or after a specified time interval. For example, time-of-day (TOD) rules can:

- Perform the command sequences necessary for shift changes.
- Periodically check samples of system information.
- Check the availability of subsystems and started tasks.

A rule responds to a TOD event when z/OS timer associated with event expires. (Unicenter CA-OPS/MVS maintains a list of pending TOD events.) The time or date (or both) that you specify in a rule's event definition section determines when z/OS timer expires.

Use this format when coding the TOD event definition section:

```
)TOD todspec1 [,interval] [,endtodspec] [,maxexecs]
[,msgallow][,catchval] [,synch]
[todspec2 [,interval] [,endtodspec] [,maxexecs]
[todspec3 [,interval] [,endtodspec] [,maxexecs]
 . . .
[todspec10 [,interval] [,endtodspec] [,maxexecs]
```

Follow these guidelines when specifying the todspec value:

- You can specify a date or time, or both, in any order.

  If you omit the date —The rule executes every day.

  If you omit the time —The rule executes at midnight.

- Note: If you specify both a date and a time, separate them with a space.
- To instruct a rule to execute after enablement at a specified interval, you can use this format for the todspec value:

  *+nn value

  where nn is a number greater than 0, and value is DAY(S), WEEK(S), HOUR(S), MINUTE(S) or MIN(S), or SECOND(S) or SEC(S). For example, the following rule executes 1 minute after it is enabled and every 30 seconds thereafter, until it executes a total of three times:

  ```
  )TOD *+1 MINUTE,30 SECONDS,,3
  ```

  Uppercase or lowercase letters are acceptable.

  The event definition specifier can contain up to ten todspec values.

  Note: If you specify more than one todspec value, place each one on a separate line.

In addition to the date and time, you can specify one or more optional qualifiers to further define your todspec TOD specifier.

Note: Although you can specify the msgallow, catchval, and synch qualifiers only once within a rule, you may specify them on any line of the TOD event definition section. You do not have to specify these three qualifiers on the same line with each other; you may specify them in any combination you wish. The values of these qualifiers apply to all todspec values in that rule.

- Interval

  The amount of time—that is, the number of specified time units—that the AOF waits between rule executions.

- endtodspec

  The ending TOD specifier. The todspec value no longer triggers the rule once the specified date and/or time has occurred.

  Note: The endtodspec value, if specified, must match the format of the todspec value.

- Maxexecs

  The maximum number of times the rule can execute. The todspec value no longer triggers the rule once this limit has been reached.

- Msgallow

  Allows or suppresses the OPS3900O message, indicating the next TOD setting for a rule. Do not specify the msgallow qualifier more than once within a rule. Valid values are:

  MSG, which allows the message

  NOMSG, which suppresses the message

  Default —MSG

- catchval

  Determines whether catch-up processing occurs for the rule. Do not specify the catchval qualifier more than once within a rule. Valid values are:

  CATCHUPYES - perform catch-up processing for this rule.

  CATCHUPNO - do not perform catch-up processing for this rule.

  CATCHUPMAN - ask the operator whether this rule requires catch-up processing.

  Default —CATCHUPNO

  Important! Neither CATCHUPYES nor CATCHUPMAN may be specified on the TOD rule specification when enabling a dynamic TOD rule. Attempting to do so causes a syntax error, and the rule is not enabled.

- Synch

  Determines whether TOD rule execution is synchronized on a rounded interval boundary. Do not specify the synch qualifier more than once within a rule. Valid values are:

  SYNCH - rule execution is synchronized.

  NOSYNCH - rule execution is not synchronized.

  Default —SYNCH

  The SYNCH value does not apply to TOD rules that specify a time to execute after enablement (that is, rules in which the *+nn value format is used). These rules always execute with a NOSYNCH value.

  Important! The optional todspec qualifiers are position-dependent. If you want to omit a qualifier but want to specify the next one, insert a place-holding comma in place of the omitted qualifier.

The OPS/REXX RETURN statement has no special meaning in the processing section of a time-of-day rule. The return value has no effect on AOF processing.

When coding the event definition section of a TOD rule, you can specify a date or time, or both (in any order). Specifying an interval is optional.

Date TOD Specifier

Any of these formats is acceptable for specifying the date specifier:

- dd MMM year
- yy/mm/dd
- weekday:

- dd

  A two-digit integer (01 through 31) corresponding to a day of the month.
- MMM

  One of the following three-character abbreviations for a month: JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, or DEC.
- year

  A four-digit year (for example, 2003).

  Note: The first two digits of year (indicating the century) are optional. For example, you can abbreviate 1999 to 99.
- Yy

  A two-digit integer (for example, 03) corresponding to a year.
- mm

  A two-digit integer (01 through 12) corresponding to a month of the year.
- weekday

  The full name of a weekday (for example, SUNDAY, MONDAY).

Time TOD Specifier

Use this format for specifying the time specifier:

```
hh:mm[:ss]
```

- hh A two-digit integer (00 through 23) indicating the hour.
- mm A two-digit integer (00 through 59) indicating the minutes after the hour.
- ss A two-digit integer (00 through 59) indicating the seconds after the minute. This value is optional.

Interval TOD Specifier

Use this format for specifying the interval specifier:

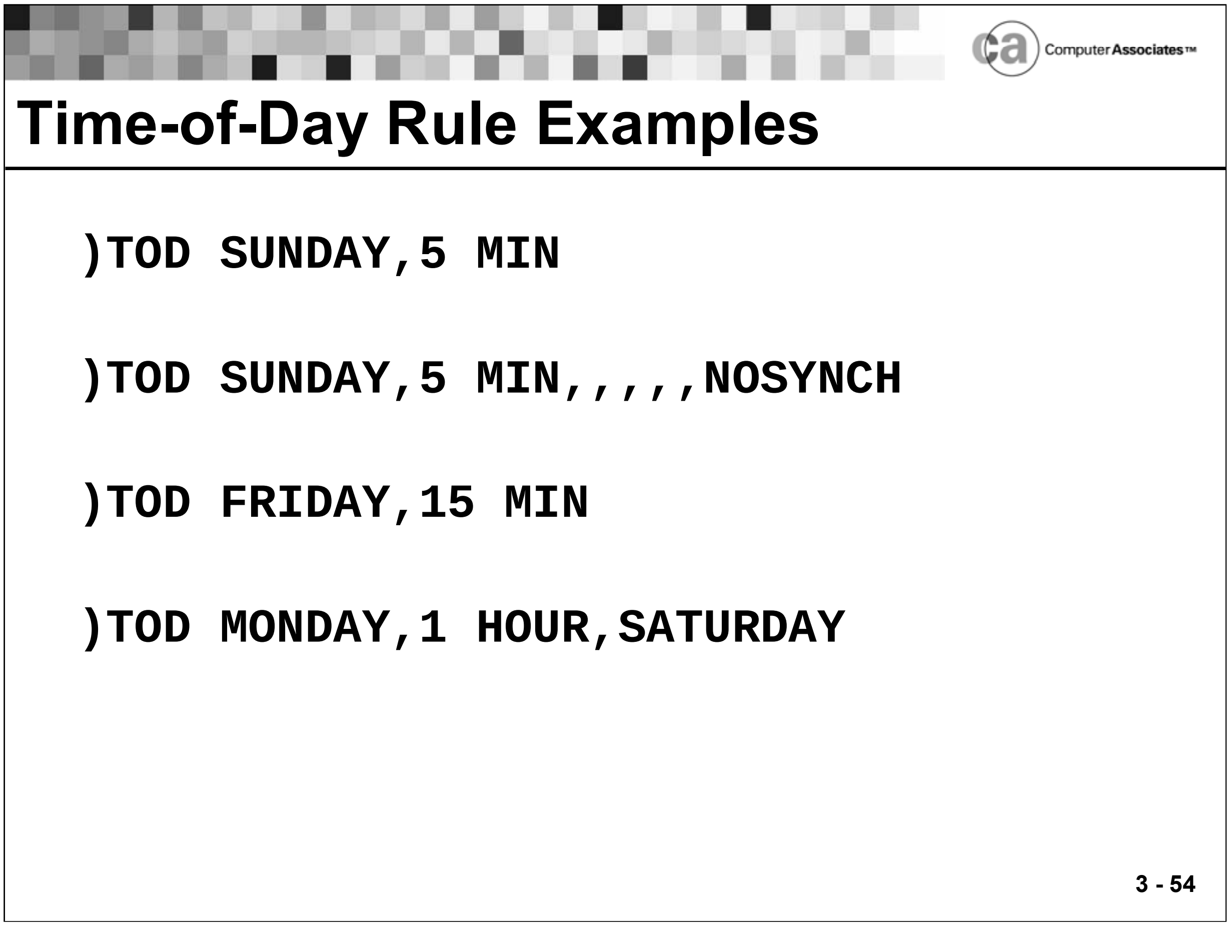
```
n interval
```

Note: You can also use the 24-hour military time format described earlier in this section.

- n An integer multiplier indicating the number of interval time units interval.

  One of the following time units: DAY(S), WEEK(S), HOUR(S), MINUTE(S) or MIN(S), SECOND(S) or SEC(S).

  Examples - 2 WEEKS, 30 SECS, 1 MIN, 30 DAYS



Time-of-Day Rule Examples

```
)TOD SUNDAY,5 MIN

)TOD SUNDAY,5 MIN,,,,,NOSYNCH

)TOD FRIDAY,15 MIN

)TOD MONDAY,1 HOUR,SATURDAY
```

3 - 54

Time-of-Day Rule Examples

Below are examples of various TOD rules. In the following examples, a week runs from Sunday to Saturday.

- This rule executes every day of the week, every five minutes, on a five-minute boundary from midnight:

  ```
  )TOD SUNDAY,5 MIN
  ```

- This rule executes every day, every five minutes, on a five-minute boundary from the time the rule is enabled:

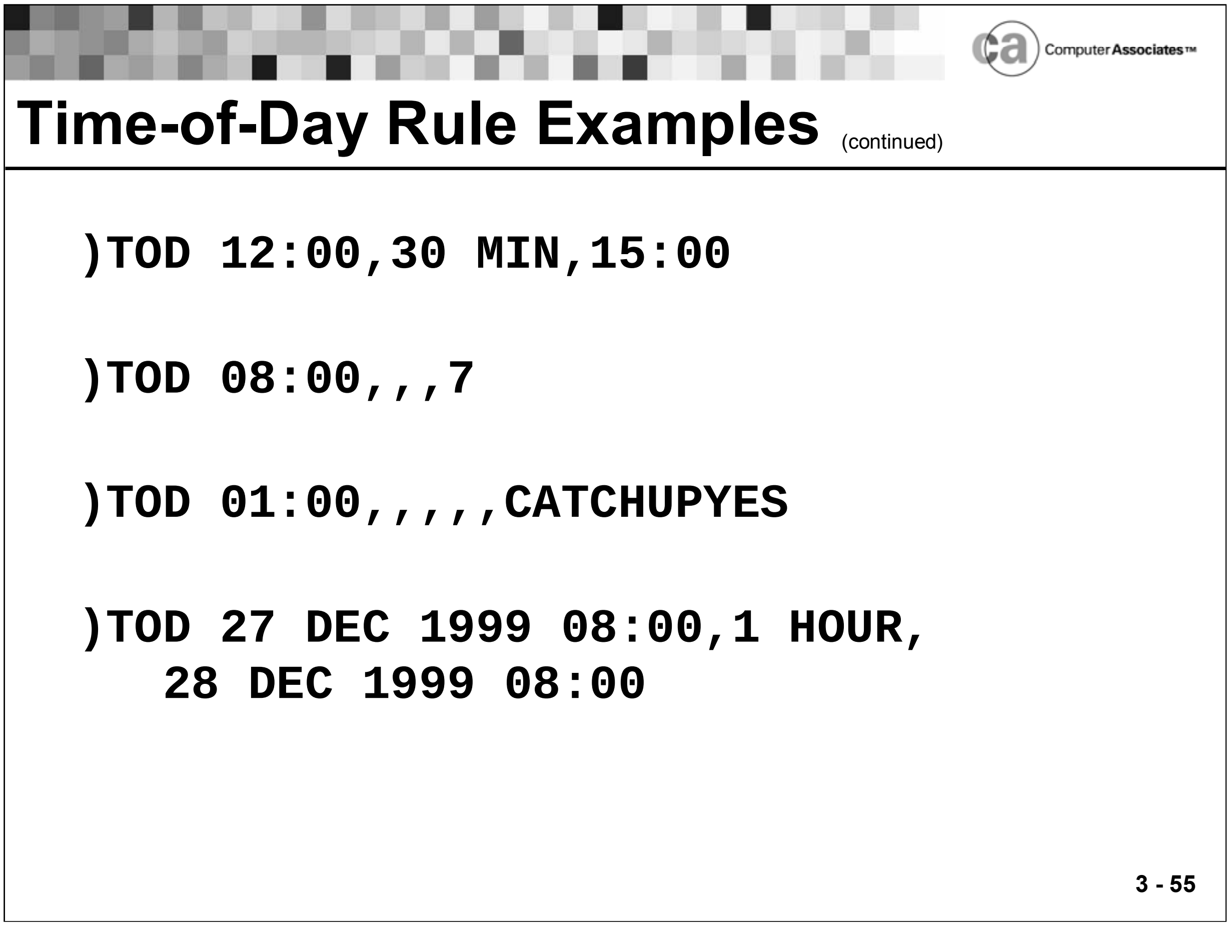
  ```
  )TOD SUNDAY,5 MIN,,,,,NOSYNCH
  ```

- This rule executes every 15 minutes on Friday and Saturday:

  ```
  )TOD FRIDAY,15 MIN
  ```

- This rule executes every hour, Monday through Friday:

  ```
  )TOD MONDAY,1 HOUR,SATURDAY
  ```



Time-of-Day Rule Examples (continued)

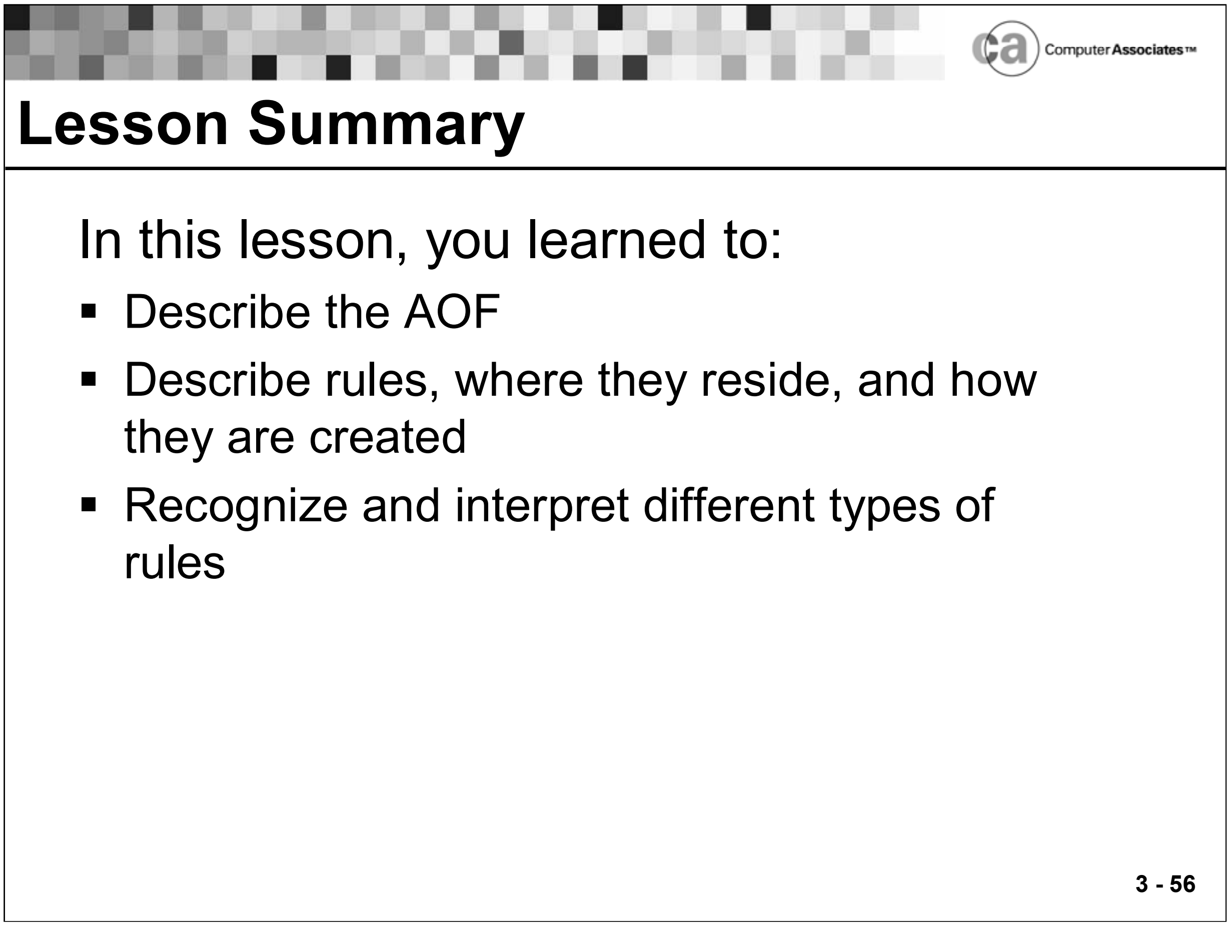
```
)TOD 12:00,30 MIN,15:00

)TOD 08:00,,,7

)TOD 01:00,,,,,CATCHUPYES

)TOD 27 DEC 1999 08:00,1 HOUR,
   28 DEC 1999 08:00
```

3 - 55

Time-of-Day Rule Examples

Below are more examples of TOD rules. Remember that a week runs from Sunday to Saturday.

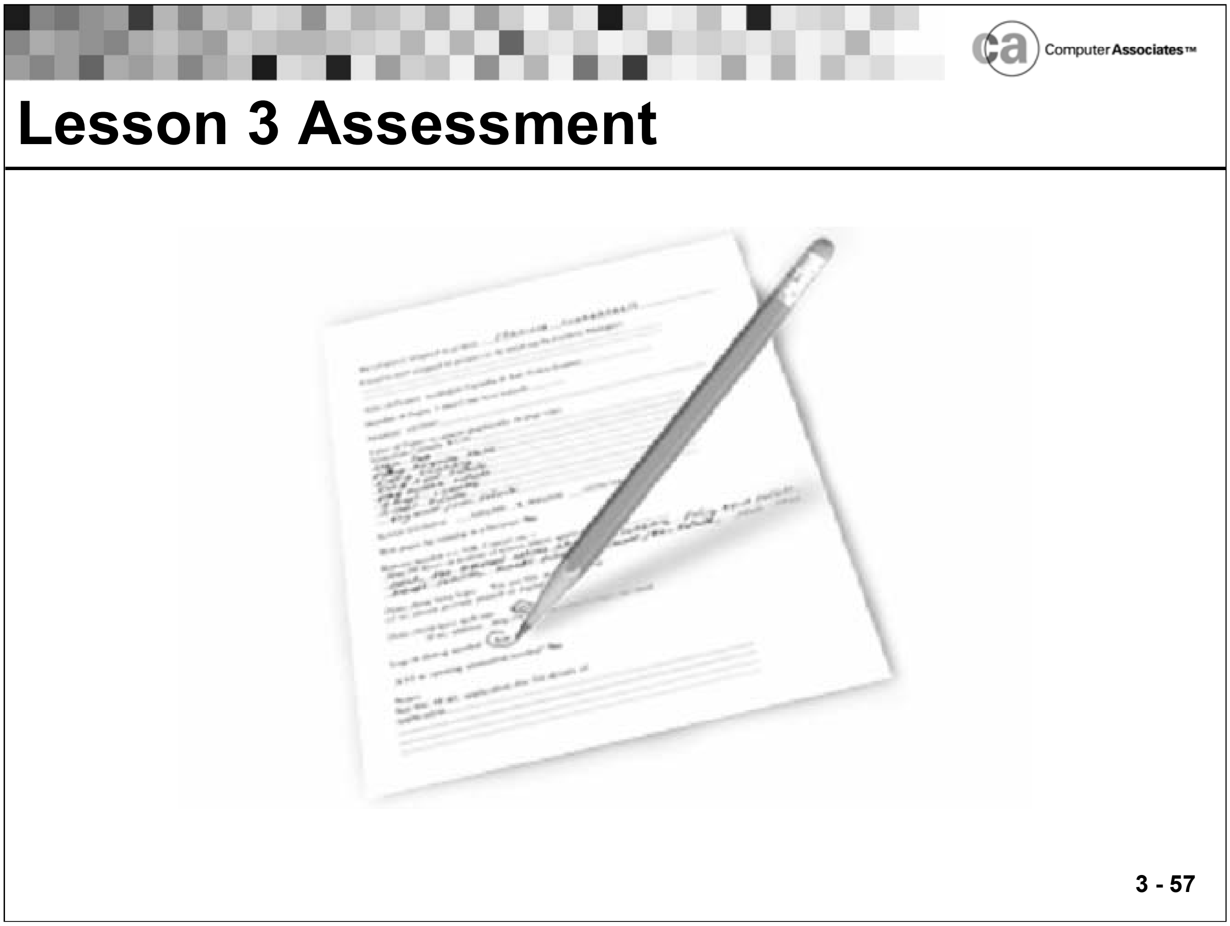
- This rule executes every half hour from 12:00 to 15:00, every day:
  ```
  )TOD 12:00,30 MIN,15:00
  ```
- This rule executes every day at 8:00 for the next seven days:
  ```
  )TOD 08:00,,,7
  ```
- This rule executes every day at 01:00 and it will catch up if Unicenter CA-OPS/MVS is down:
  ```
  )TOD 01:00,,,,,CATCHUPYES
  ```
- This rule executes from 8:00 December 27 through 7:00 December 28:
  ```
  )TOD 27 DEC 1999 08:00,1 HOUR,  28 DEC 1999 08:00
  ```



Lesson Summary

In this lesson, you learned to:

- Describe the AOF
- Describe rules, where they reside, and how they are created
- Recognize and interpret different types of rules



Lesson 3 Assessment

3 - 57

Lesson 3 Assessment

Note: A week runs from Sunday to Saturday.

1. True/False: AOF rules contain OPS/REXX programs.

____

2. When would this rule execute?)TOD 06:00,,,6

____

3. What is the major function of the AOF?

____

4. Which rule section specifies the actions the rule takes when it becomes enabled?

____

Lesson 3 Assessment (continued)

5. True/False: Each rule set is a separate data set.

____

6. When would this rule execute?)TOD 07:30,,,,,CATCHUPYES

____

7. Name three methods for writing rules.

____

8. Which rule type can be used as an effective tool for operators to perform the necessary command sequences for shift changes?

____

9. Which rule section specifies the system event that causes the rule to execute?

____

10. True/False: A rule must contain an)END statement.

____

11. How does the AOF recognize and respond to events?

____

12. Which rule type enables you to create a new command?

____

13. When would this rule execute?)TOD 02:00

____

14. Which rule section specifies the action the rule takes when it is disabled?

____

Lesson 3 Assessment (continued)

15. True/False: Rules always execute in a predictable order.

16. Which part of a rule marks the end of the rule?

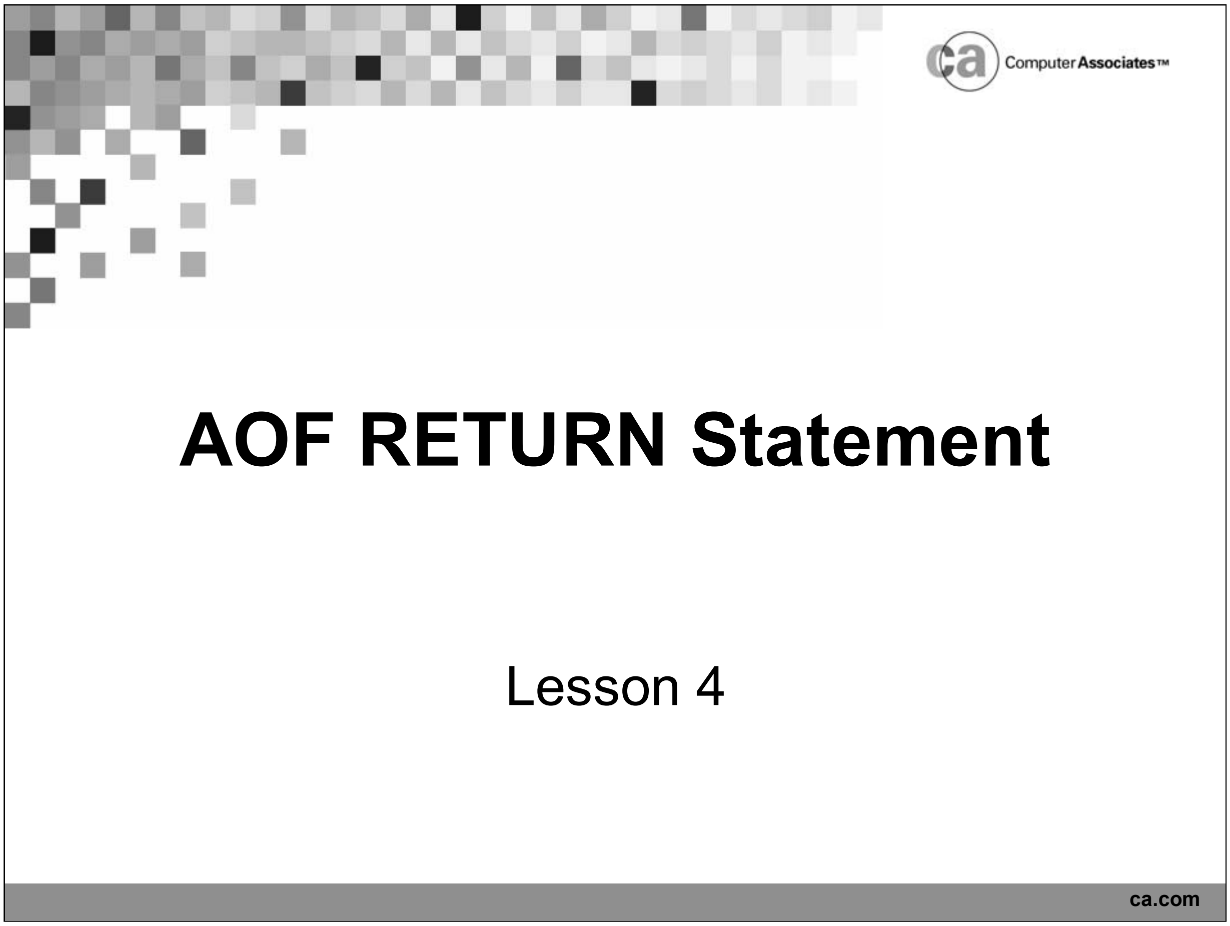
17. Which rule section specifies the action to take in response to the system event that triggered the rule?

18. True/False: Rules are stored within rule sets.

19. Which rule type enables you to respond to a WTOR message?

20. When would this rule execute?)TOD *+2 MINUTES, 15 SECONDS,,5

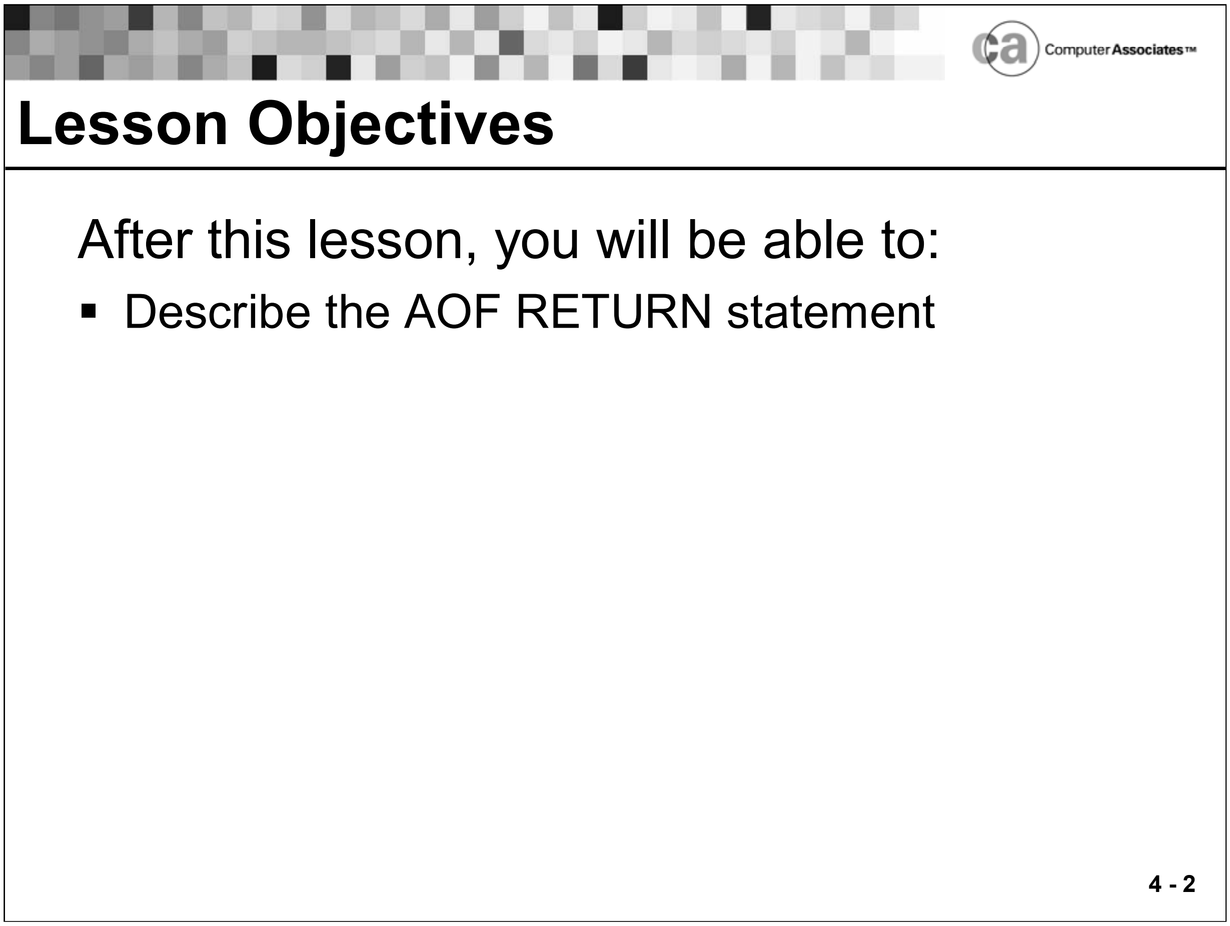
Notes:



AOF RETURN Statement

Lesson 4

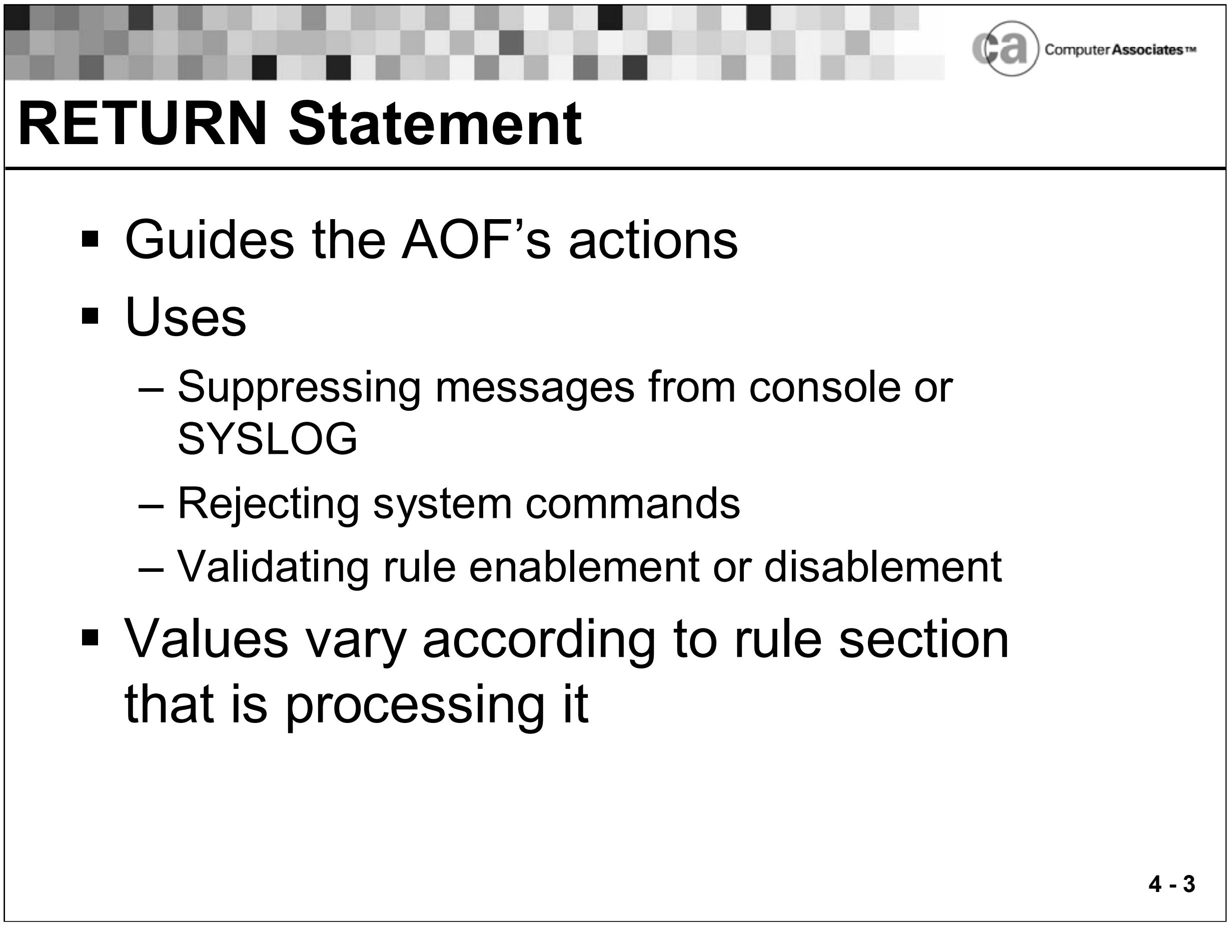
ca.com



Lesson Objectives

After this lesson, you will be able to:

- Describe the AOF RETURN statement



RETURN Statement

- Guides the AOF’s actions
- Uses
  - Suppressing messages from console or SYSLOG
  - Rejecting system commands
  - Validating rule enablement or disablement
- Values vary according to rule section that is processing it

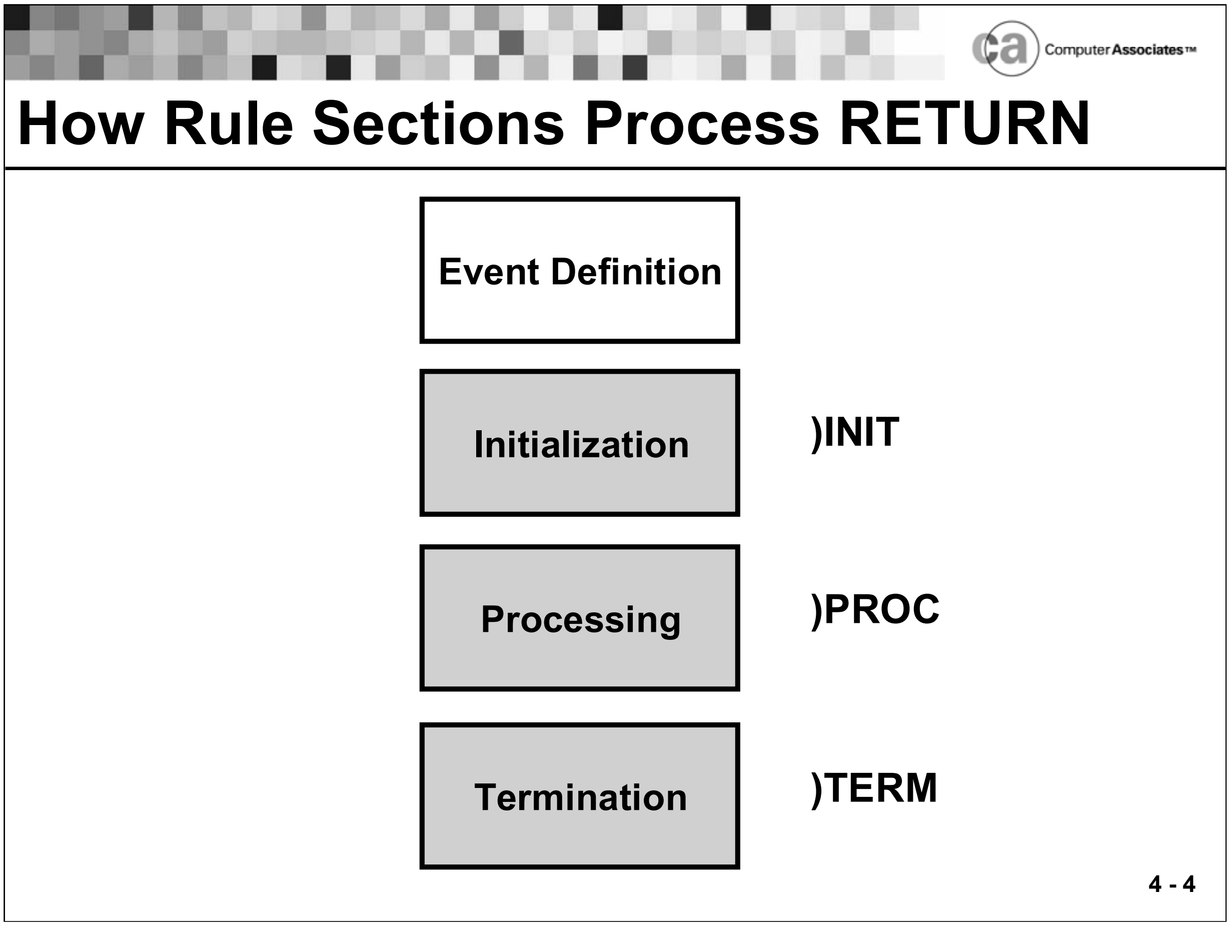
4 - 3

RETURN Statement

The AOF RETURN statement allows you to:

- Create logic in AOF rules to suppress messages from appearing on the console or in SYSLOG.
- Reject system commands.
- Validate rule enablement or disablement.

OPS/REXX fully supports the RETURN statement. AOF RETURN statement values vary according to the rule section that is processing the statement. The)INIT,)TERM, and)PROC sections of AOF rules are OPS/REXX programs called by the AOF. The values returned by these rule sections (via the RETURN statement) guide the AOF’s actions. For example, if a)INIT section returns “REJECT,” the rule is neither enabled nor executed.



How Rule Sections Process RETURN

Event Definition

Initialization)INIT

Processing)PROC

Termination)TERM

4 - 4

How Rule Sections Process the RETURN Statement

)INIT Section

- The RETURN statement in the)INIT section of an AOF rule allows or disallows a rule’s enablement. Creating logic to prevent a rule from being enabled may be necessary in an environment in which you have multiple Unicenter CA-OPS/MVS images sharing rule sets and you only want particular rules to be enabled on certain systems.
- Valid values for a RETURN statement in a rule’s initialization section are:

  ACCEPT

  Allows the rule to be enabled.

  REJECT

  Prevents the rule from being enabled.
- If you do not specify a return value, the default value is ACCEPT.
- If a run-time error occurs, the return value is ACCEPT (assuming that the error occurs while the initialization section is executing).

)PROC Section

- The RETURN statement in the)PROC section of an AOF rule affects the disposition of the event. It works differently according to the type of event the rule is acting upon. That is, the RETURN statement in a)MSG rule has different values than those in a)CMD rule.
- Valid RETURN statement values for the)PROC section of each rule type are identified in the OPS/MVS documentation. It is important to understand the effect of each value before writing any of the AOF rule types.

)TERM Section

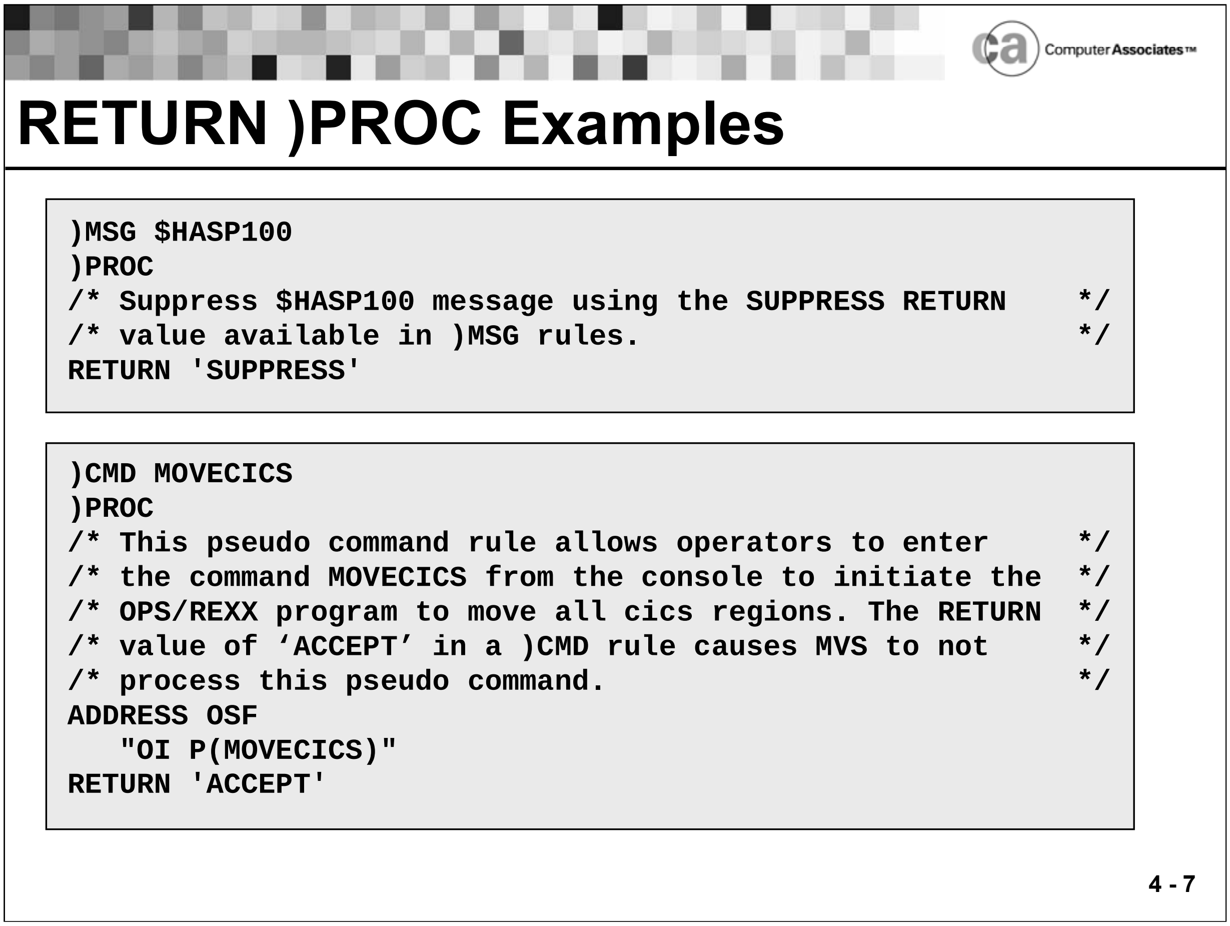
- The RETURN statement in the)TERM section of an AOF rule allows or disallows a rule's disablement.
- Valid values for a RETURN statement in a rule's termination section are:

  ACCEPT

  Allows the rule to be disabled

  REJECT

  Prevents the rule from being disabled
- If you do not specify a return value, the default value is ACCEPT.
- If a run-time error occurs, the return value is ACCEPT (assuming that the error occurs while the termination section is executing).
- Note: A return value of REJECT stops the disablement of a single rule only. If you disable a rule set, the enabled rules within the rule set are always disabled, regardless of the RETURN values in the individual rules.
- Examples for each rule section follow.

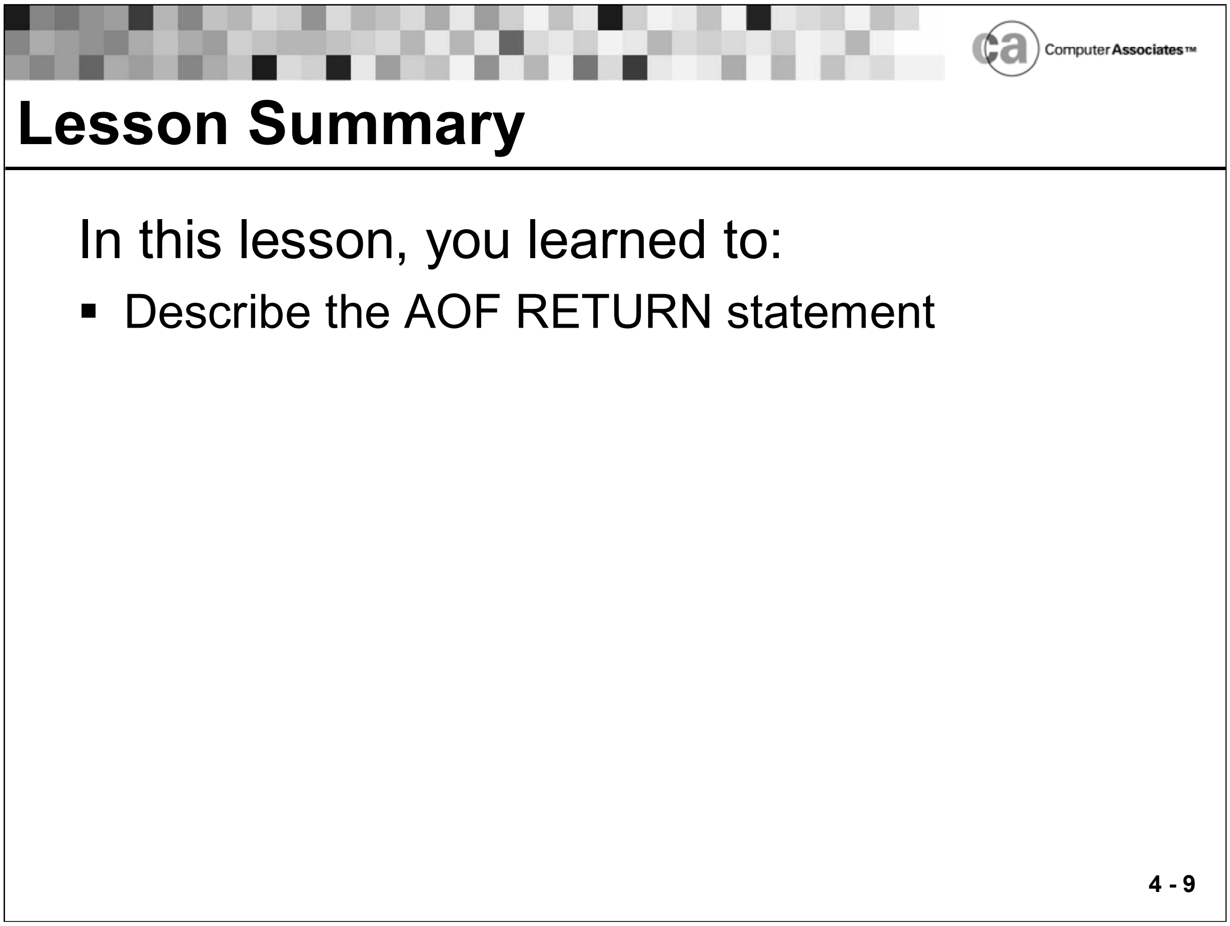


RETURN)PROC Examples

```
)MSG $HASP100
)PROC
/* Suppress $HASP100 message using the SUPPRESS RETURN    */
/* value available in )MSG rules.                         */
RETURN 'SUPPRESS'
```

```
)CMD MOVECICS
)PROC
/* This pseudo command rule allows operators to enter     */
/* the command MOVECICS from the console to initiate the  */
/* OPS/REXX program to move all cics regions. The RETURN  */
/* value of ‘ACCEPT’ in a )CMD rule causes MVS to not     */
/* process this pseudo command.                           */
ADDRESS OSF
   "OI P(MOVECICS)"
RETURN 'ACCEPT'
```

4 - 8



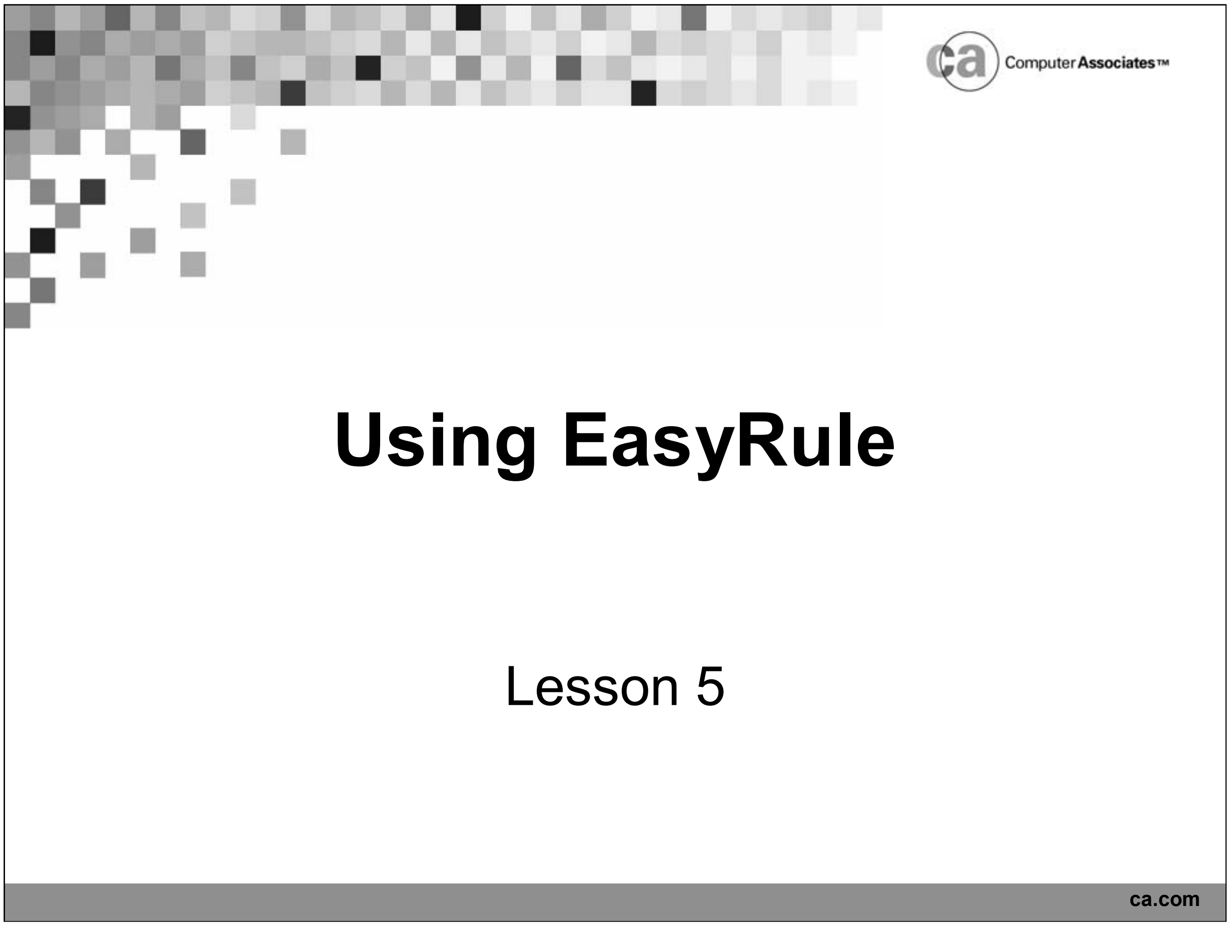
Lesson Summary

In this lesson, you learned to:

- Describe the AOF RETURN statement

4 - 9

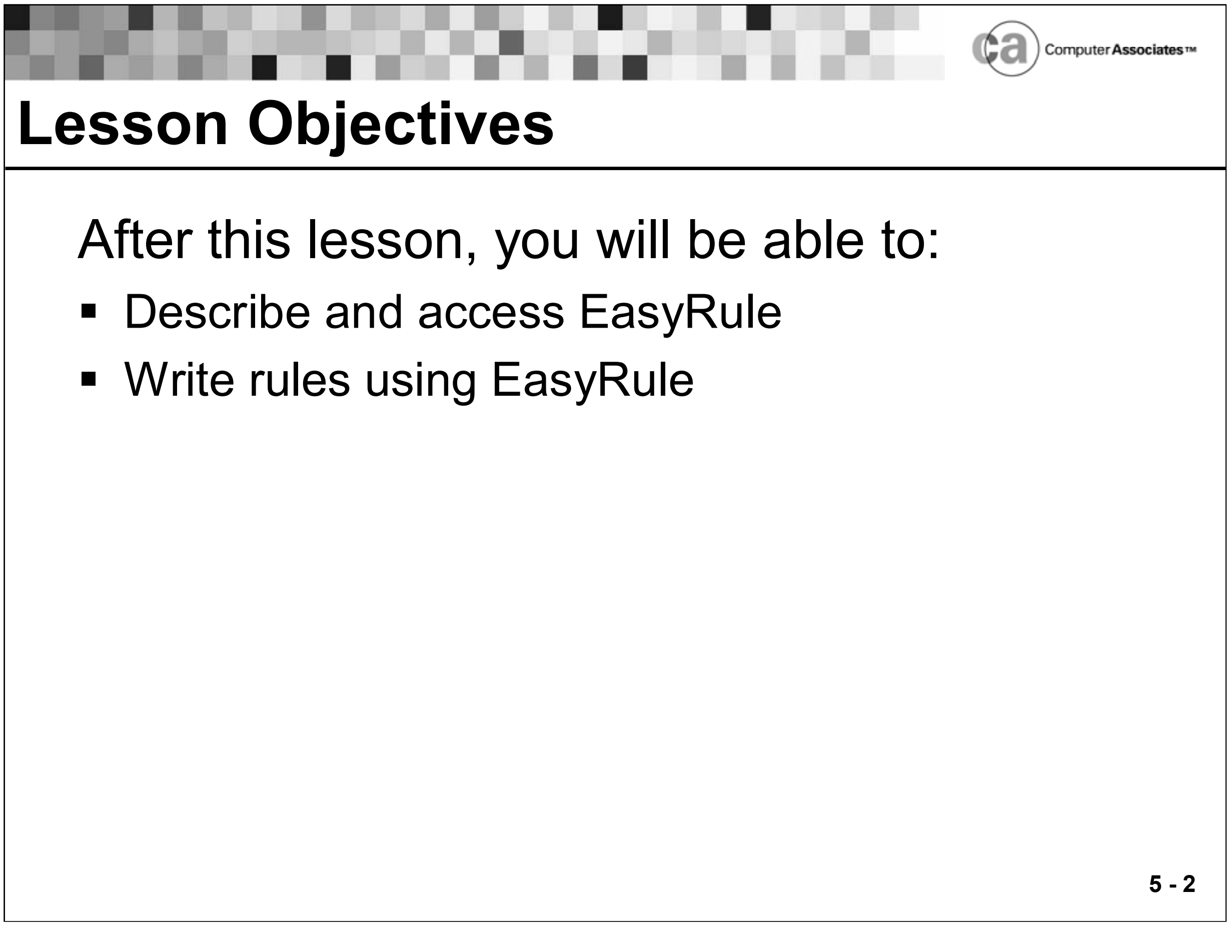
Notes:



Using EasyRule

Lesson 5

ca.com



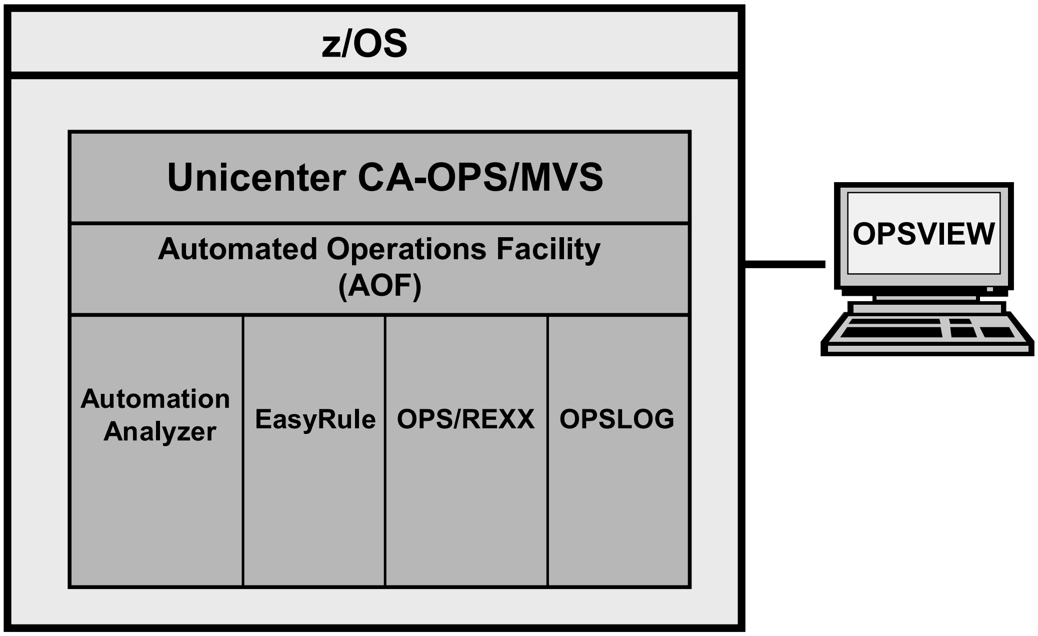
Lesson Objectives

After this lesson, you will be able to:

- Describe and access EasyRule
- Write rules using EasyRule

5 - 2

EasyRule



z/OS

Unicenter CA-OPS/MVS

Automated Operations Facility (AOF)

Automation Analyzer

EasyRule

OPS/REXX

OPSLOG

OPSVIEW

EasyRule

- EasyRule is a user-friendly, panel-driven facility that walks you through the AOF rule creation and modification processes. It enables you to build Unicenter CA-OPS/MVS rules without the need for programming.
- EasyRule is an online OPSVIEW facility that offers a “fill-in-the-blanks” approach to building a rule. EasyRule not only makes generating rules fast and easy, but 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 that is generated by EasyRule contains descriptive REXX comments that describe the conditions and actions that are 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.
- To use EasyRule, you must be familiar with ISPF line editing and navigation, and you must understand how the console works.

How Does EasyRule Build 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. Your entries will include such information as events, conditions, and actions that affect the rule's execution.
- When you finish making entries on the panels, EasyRule generates the rule as OPS/REXX code and retains it in memory. (EasyRule generates OPS/REXX code in mixed case.)
- 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.
- If you choose to save the rule, you can then enable, test, and disable it just as you would any other rule. Finally, you can move the rule to the production environment.

How Will EasyRule Benefit Me?

- If you are a novice user, you can use EasyRule to generate most of the automation you need with just 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 ISPF's editing tools to add more complex logic to the rule.
- The code that EasyRule generates is clean and efficient. Therefore, if your interests are in learning how to write OPS/REXX code, you can browse the rules EasyRule generates to learn about the OPS/REXX language.

Guidelines for 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 a particular rule, you cannot use EasyRule to modify it.
- If you wish, you can select EasyRule's automatic step-through feature, which takes you from one fill-in-the-blanks panel to the next without your having to make menu selections.

- After you access EasyRule, you must decide how you want to proceed: manually or automatically. The EasyRule Primary panel prompts you for your choice. If you want to move through EasyRule's 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.
- Anytime that you want information while using EasyRule, such as an example or the definition of an unfamiliar term, you can press PF1/13 to access EasyRule help.
- Do not imbed syntax for REXX comments (for example, /* comment-text */) within your panel entries.

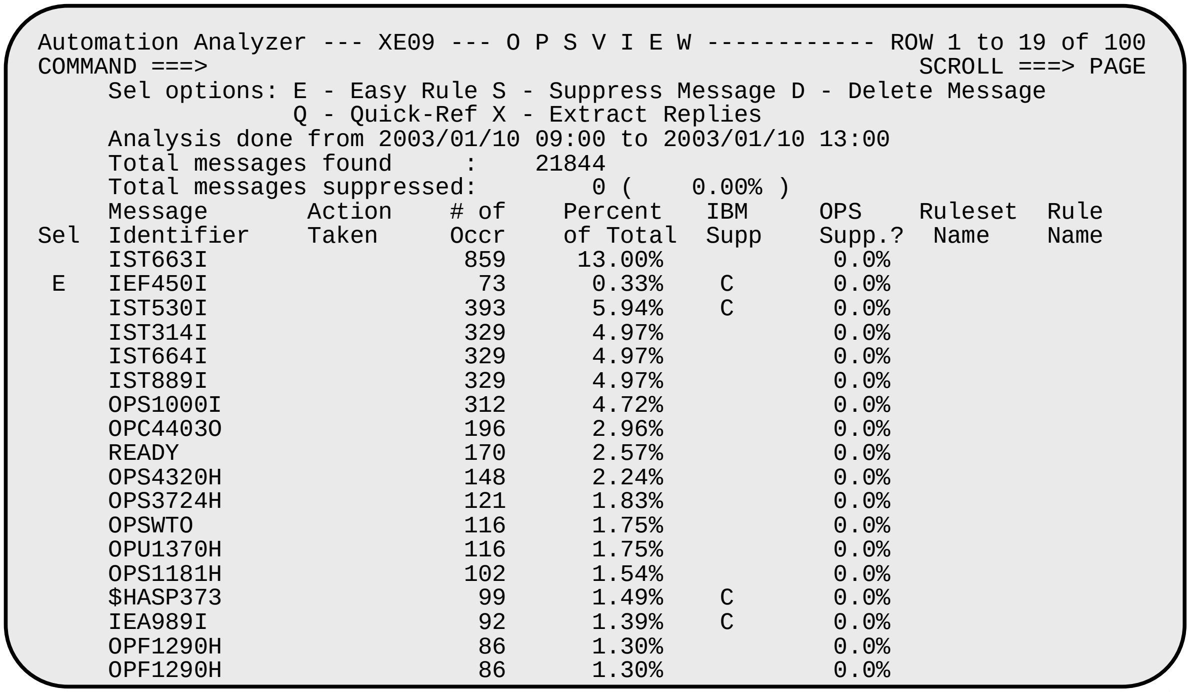
About EasyRule Panels

- To help you to build a rule, EasyRule presents you with a series of panels that are 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. For example, if you are creating an OMEGAMON rule, EasyRule presents you with a panel that prompts you to specify an OMEGAMON exception ID; if you are creating a message rule, a panel that prompts you for a message ID appears instead.

Accessing More Information About a Panel

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

Accessing EasyRule



```
Automation Analyzer --- XE09 --- O P S V I E W ----------- ROW 1 to 19 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 2003/01/10 09:00 to 2003/01/10 13:00
     Total messages found     :    21844
     Total messages suppressed:        0 (    0.00% )
     Message       Action    # of    Percent   IBM     OPS    Ruleset  Rule
Sel  Identifier    Taken     Occr    of Total  Supp    Supp.?  Name    Name
     IST663I                  859     13.00%             0.0%
 E   IEF450I                   73      0.33%    C        0.0%
     IST530I                  393      5.94%    C        0.0%
     IST314I                  329      4.97%             0.0%
     IST664I                  329      4.97%             0.0%
     IST889I                  329      4.97%             0.0%
     OPS1000I                 312      4.72%             0.0%
     OPC4403O                 196      2.96%             0.0%
     READY                    170      2.57%             0.0%
     OPS4320H                 148      2.24%             0.0%
     OPS3724H                 121      1.83%             0.0%
     OPSWTO                   116      1.75%             0.0%
     OPU1370H                 116      1.75%             0.0%
     OPS1181H                 102      1.54%             0.0%
     $HASP373                  99      1.49%    C        0.0%
     IEA989I                   92      1.39%    C        0.0%
     OPF1290H                  86      1.30%             0.0%
     OPF1290H                  86      1.30%             0.0%
```

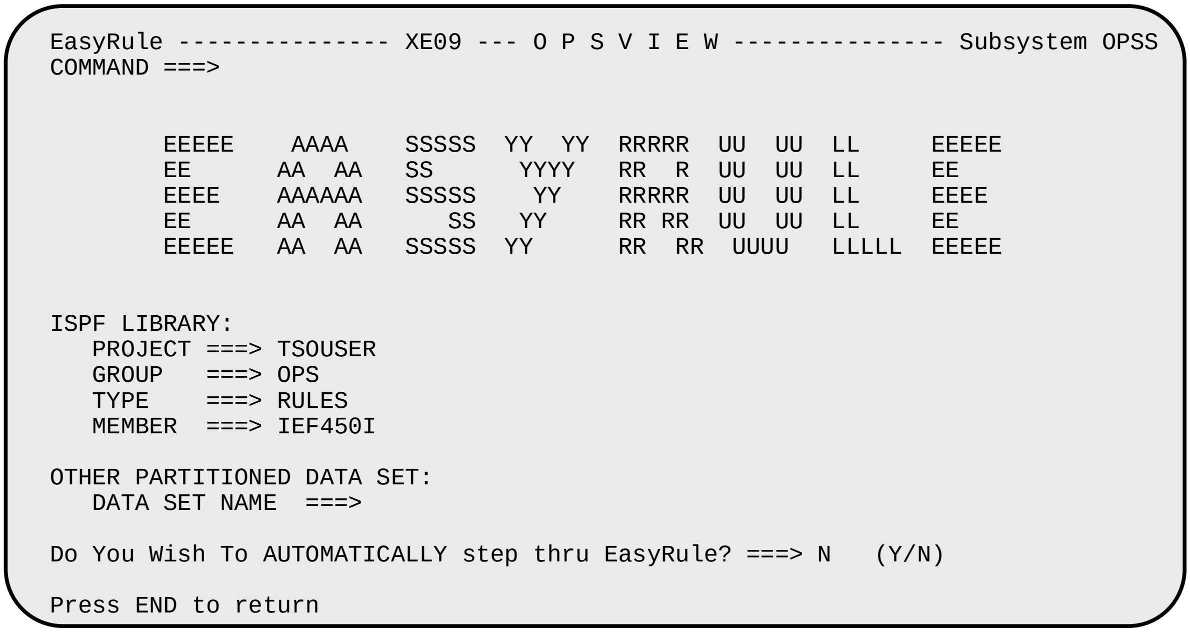
5 - 6

Accessing EasyRule

You can access EasyRule in any of the following ways:

- Type 3 in the Option field on the Editors menu and then press Enter.
- Use the ISPF jump function by typing =2.3 into any valid field within OPSVIEW and then pressing Enter.
- From the AOF TEST Rule List panel (OPSVIEW option 2.1) or the AOF CTRL Rule List panel (OPSVIEW option 4.5.1):
  - To create a new rule, enter the EASYRULE primary command in the Command field.
  - To modify an existing rule, type R next to the name of the rule you want to modify and press Enter.
- 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; then press Enter.

Primary Panel



```
EasyRule --------------- XE09 --- 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 ===> TSOUSER
   GROUP   ===> OPS
   TYPE    ===> RULES
   MEMBER  ===> IEF450I

OTHER PARTITIONED DATA SET:
   DATA SET NAME  ===>

Do You Wish To AUTOMATICALLY step thru EasyRule? ===> N   (Y/N)

Press END to return
```

Primary Panel

After you access EasyRule using one of the methods described previously, you will see a display similar to the one shown above.

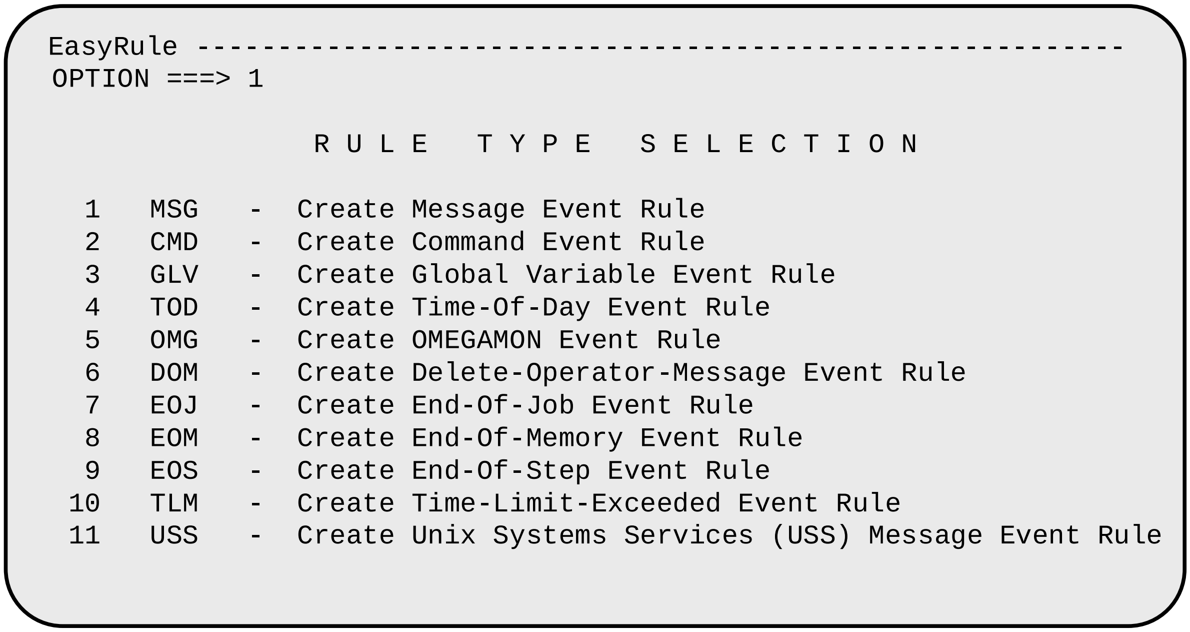
Before you can create or modify a rule, you must tell EasyRule the name of the rule set that will contain (or already contains) the rule. You will use the Project, Group, and Type fields to do so:

- **`PROJECT ===> ruleprefix`**
- **`GROUP   ===> rulesetname`**
- **`TYPE    ===> rulesuffix`**

You must also specify a new or existing member name in the member field. Each member of a rule set contains a single rule.

Action: Complete the panel as shown above and then press Enter. (In place of TSOUSER, type the user ID given to you by your instructor.) The Rule Type Selection panel appears.

Select Rule Type



```
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 Systems Services (USS) Message Event Rule
```

Select Rule Type

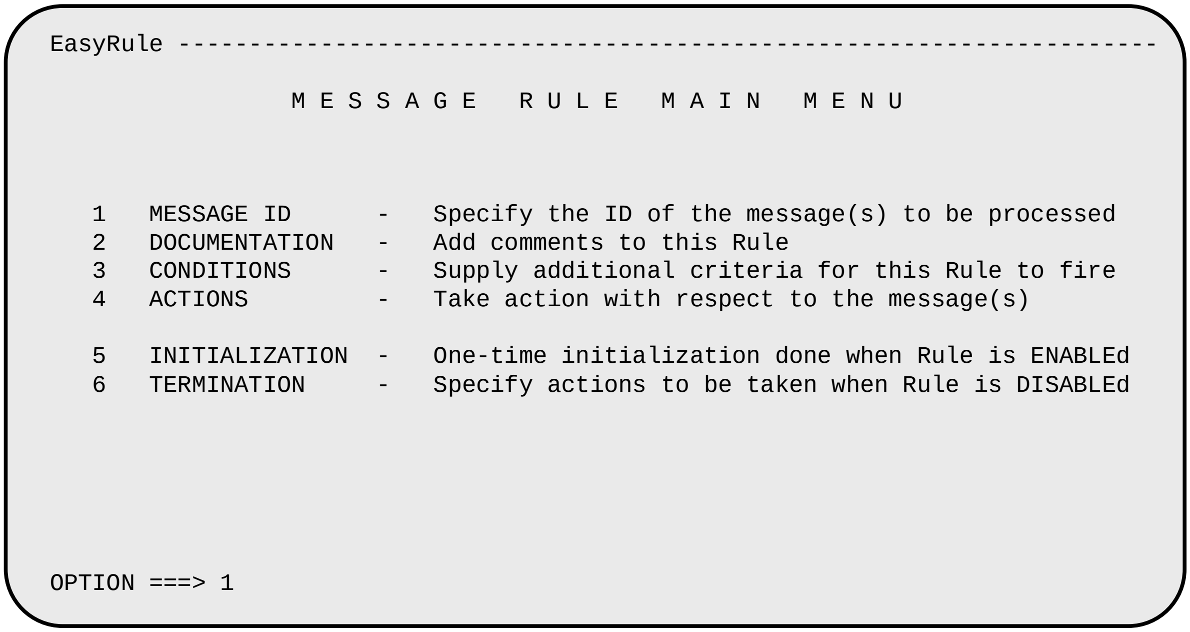
When you specify the member name for a new rule on the EasyRule Primary Panel, the Rule Type Selection panel appears, shown above.

Note: 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.

To select a rule type, enter its code into the Option field.

Action: For our example, we will be creating a message rule. Type a 1 in the Option field and then press Enter. The Message Rule Main Menu displays.

Main Menu



```
EasyRule -----------------------------------------------------------------

                 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






OPTION ===> 1
```

5 - 9

Main Menu

EasyRule provides a main menu for each type of rule. The main menu for message rules is shown above.

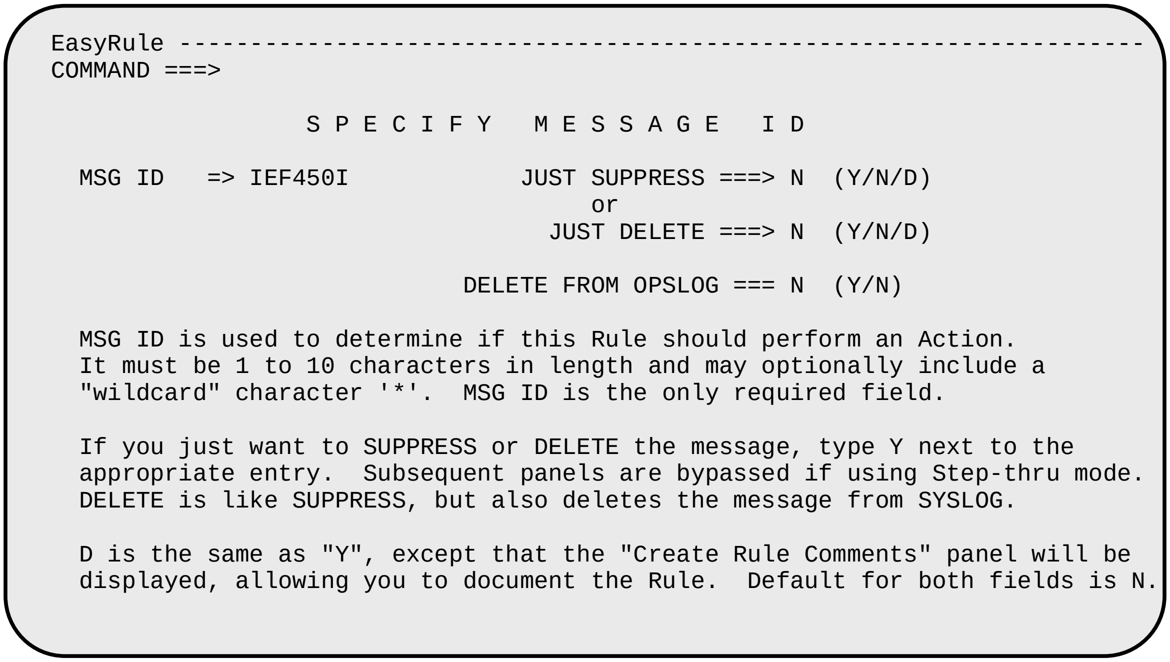
Action: We will specify a message ID next, so type a 1 in the Option field and then press Enter. The Primary Event Specification panel displays.

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:

1. 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.
2. 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.
3. Accesses a submenu from which you can choose conditions for EasyRule to use as selection criteria for the rule. The content of this submenu varies for different types of rules. These conditions are in addition to the primary selection criterion you specify with option 1.
4. Accesses a submenu 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 within option 4 will occur unconditionally. The content of this submenu is different for each type of rule.
5. Accesses a panel on which you specify what systems the rule is initialized and 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.
6. Accesses a panel on which you specify the actions Unicenter 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 Event



```
EasyRule -----------------------------------------------------------------
COMMAND ===>

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

  MSG ID   => IEF450I            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 the "Create Rule Comments" panel will be
  displayed, allowing you to document the Rule.  Default for both fields is N.
```

5 - 11

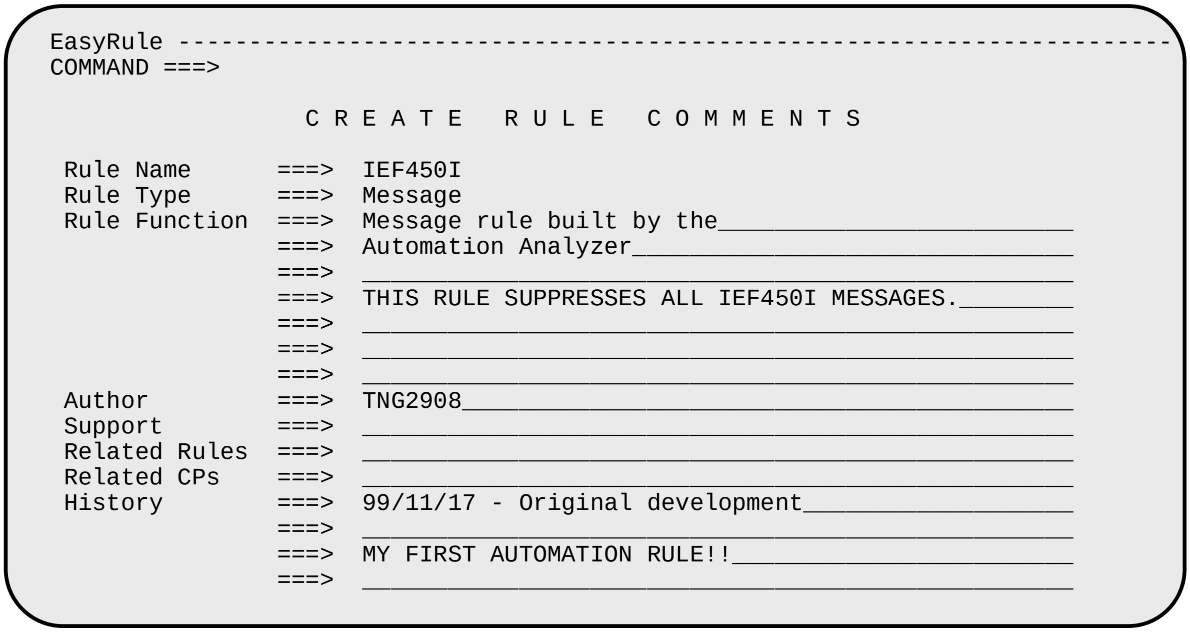
Specify Event

Use the Primary Event Specification panel to specify a primary event for your 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. A sample Primary Event Specification panel pertaining to message rules is shown above.

When EasyRule generates the OPS/REXX code for your rule, the rule's type and the rule's primary event 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. You can then press PF3 to access the EasyRule Final Options Menu, from which you can end your EasyRule session.

Action: Type IEF450I in the MSG ID field and then press Enter. You are returned to the Message Rule Main Menu.

Create Rule Comments



```
EasyRule ----------------------------------------------------------------------
COMMAND ===>

                  C R E A T E   R U L E   C O M M E N T S

 Rule Name      ===>  IEF450I
 Rule Type      ===>  Message
 Rule Function  ===>  Message rule built by the_________________________
                ===>  Automation Analyzer_______________________________
                ===>  __________________________________________________
                ===>  THIS RULE SUPPRESSES ALL IEF450I MESSAGES.________
                ===>  __________________________________________________
                ===>  __________________________________________________
                ===>  __________________________________________________
 Author         ===>  TNG2908___________________________________________
 Support        ===>  __________________________________________________
 Related Rules  ===>  __________________________________________________
 Related CPs    ===>  __________________________________________________
 History        ===>  99/11/17 - Original development___________________
                ===>  __________________________________________________
                ===>  MY FIRST AUTOMATION RULE!!________________________
                ===>  __________________________________________________
```

5 - 12

Create Rule Comments

Next, we will create comments for the rule using the Create Rule Comments panel. To access this panel, type a 2 in the Option field on the Message Rule Main Menu and then press Enter.

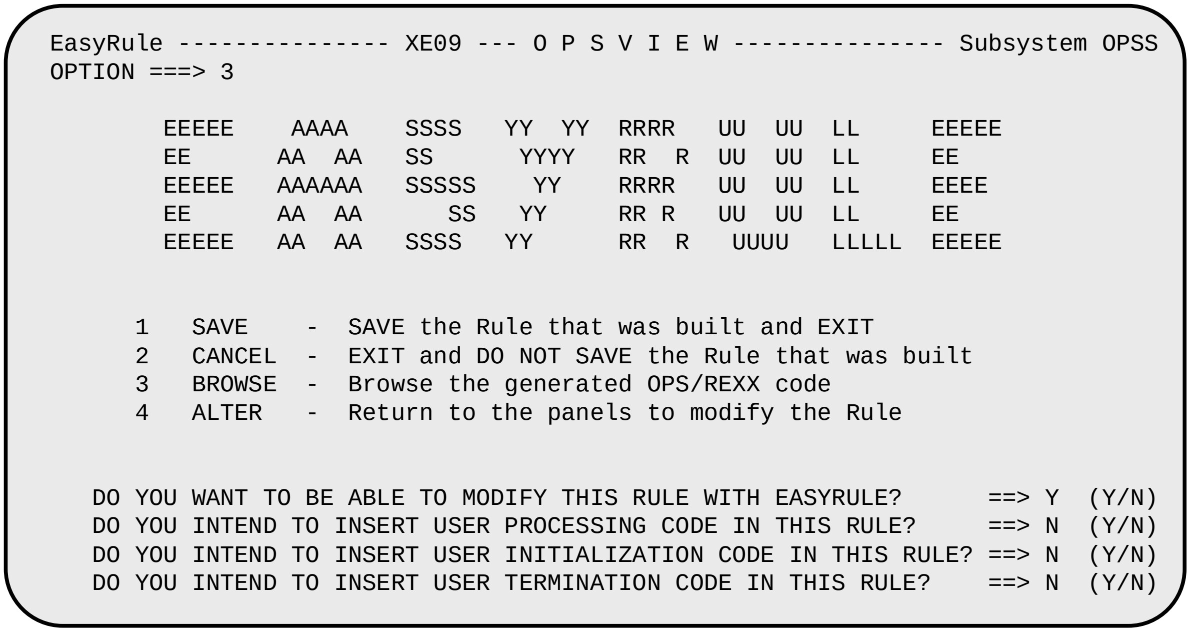
A sample panel, the same for all types of rules, is shown above.

The Create Rule Comments panel provides a structured format that you can use to create useful documentation for your 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 */) within your entries.

Action: Complete the panel as shown above and then press PF3. The Message Rule Main Menu displays.

Final Options Menu



```
EasyRule --------------- XE09 --- O P S V I E W --------------- Subsystem OPSS
OPTION ===> 3

        EEEEE    AAAA    SSSS   YY  YY  RRRR   UU  UU  LL     EEEEE
        EE      AA  AA   SS      YYYY   RR  R  UU  UU  LL     EE
        EEEEE   AAAAAA   SSSSS    YY    RRRR   UU  UU  LL     EEEE
        EE      AA  AA      SS   YY     RR R   UU  UU  LL     EE
        EEEEE   AA  AA   SSSS   YY      RR  R   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)
```

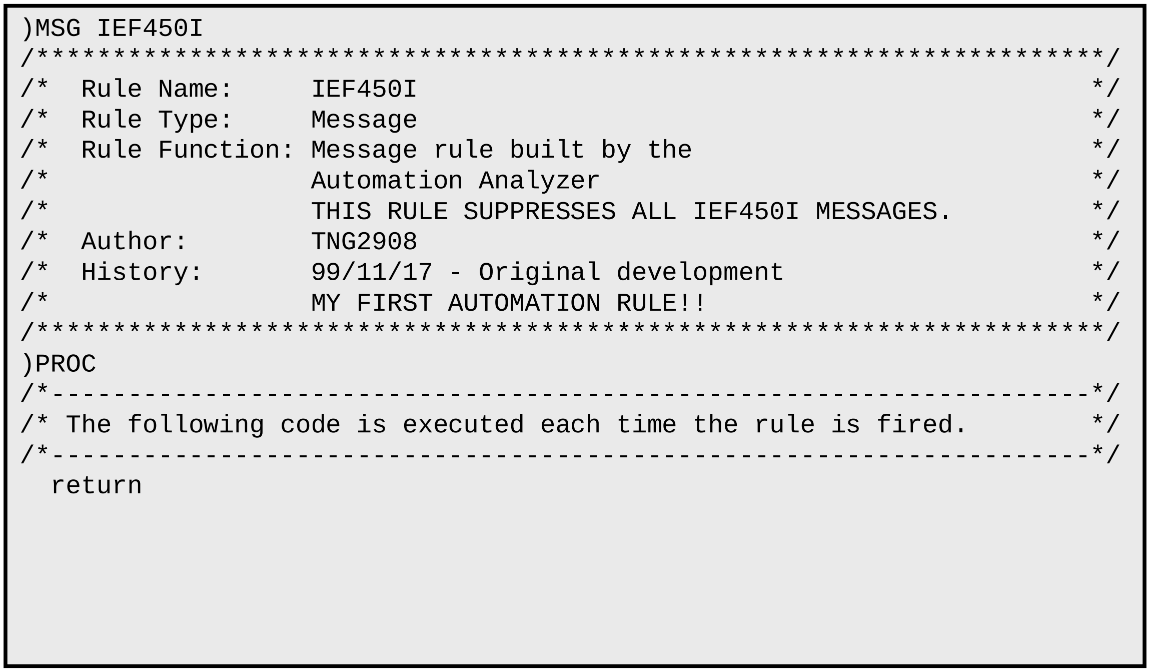
Final Options Menu

You can go to the EasyRule final options menu at any time during the rule creation or update process. This menu is automatically encountered while exiting EasyRule.

The final options menu allows you to save the rule, cancel the creation or changes made to the rule, browse the current OPS/REXX coding that has been written to the rule, or return to the primary EasyRule menu to continue with rule modifications.

The remaining options determine your ability to reopen the rule with EasyRule after saving it. Select Y on the first question if you want to be able to use EasyRule to Browse or Edit the generated REXX code. IF YOU REPLY "N", YOU CANNOT MODIFY THE GENERATED CODE VIA EASYRULE. THE ONLY WAY WILL BE TO EDIT THE MEMBER VIA ISPF. Select Y on one or more of the last three questions to reserve space for user code in the generated rule. You can enter/modify user code via ISPF and still reopen the rule via EasyRule.

View Results



```
)MSG IEF450I
/**********************************************************************/
/*  Rule Name:     IEF450I                                            */
/*  Rule Type:     Message                                            */
/*  Rule Function: Message rule built by the                          */
/*                 Automation Analyzer                                */
/*                 THIS RULE SUPPRESSES ALL IEF450I MESSAGES.         */
/*  Author:        TNG2908                                            */
/*  History:       99/11/17 - Original development                    */
/*                 MY FIRST AUTOMATION RULE!!                         */
/**********************************************************************/
)PROC
/*--------------------------------------------------------------------*/
/* The following code is executed each time the rule is fired.        */
/*--------------------------------------------------------------------*/
  return
```

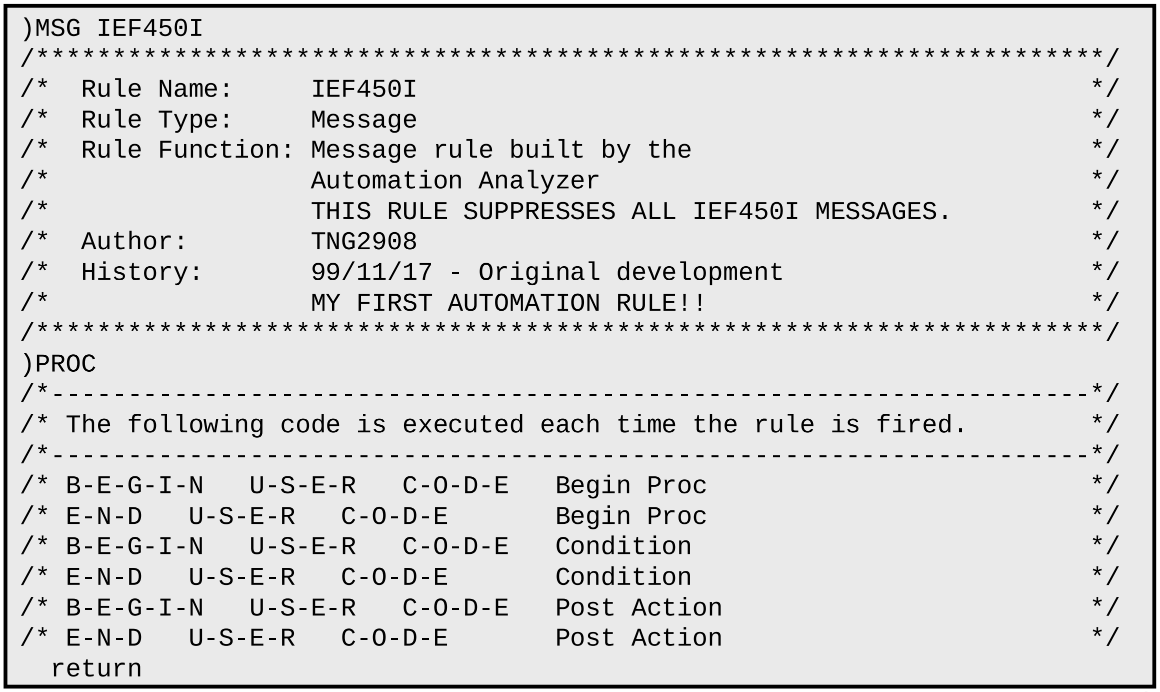
5 - 14

View Results

Above is your generated OPS/REXX code.

Action: When you are finished viewing the code, press PF3. The EasyRule Final Options Menu appears. Enter a 4 (Alter) in the Option field and then press Enter to continue writing the rule.

User Code Entry Points



```
)MSG IEF450I
/*********************************************************************/
/*  Rule Name:     IEF450I                                           */
/*  Rule Type:     Message                                           */
/*  Rule Function: Message rule built by the                         */
/*                 Automation Analyzer                               */
/*                 THIS RULE SUPPRESSES ALL IEF450I MESSAGES.        */
/*  Author:        TNG2908                                           */
/*  History:       99/11/17 - Original development                   */
/*                 MY FIRST AUTOMATION RULE!!                        */
/*********************************************************************/
)PROC
/*-------------------------------------------------------------------*/
/* The following code is executed each time the rule is fired.      */
/*-------------------------------------------------------------------*/
/* B-E-G-I-N   U-S-E-R   C-O-D-E   Begin Proc                        */
/* E-N-D   U-S-E-R   C-O-D-E       Begin Proc                        */
/* B-E-G-I-N   U-S-E-R   C-O-D-E   Condition                         */
/* E-N-D   U-S-E-R   C-O-D-E       Condition                         */
/* B-E-G-I-N   U-S-E-R   C-O-D-E   Post Action                       */
/* E-N-D   U-S-E-R   C-O-D-E       Post Action                       */
  return
```

5 - 15

User Entry Code Points

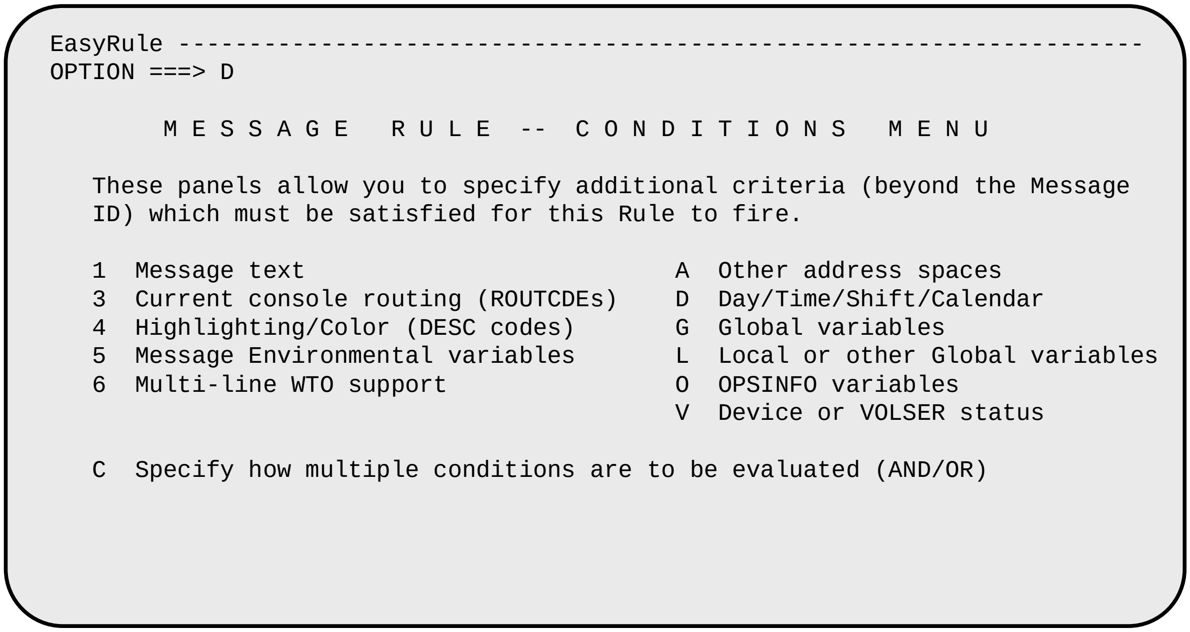
The above example shows the User Code Entry Points that are inserted if you answer Y to the question

DO YOU INTEND TO INSERT USER PROCESSING CODE IN THIS RULE? ==> N (Y/N)

that appears on the EasyRule final options menu.

Normally, if user code is entered added to a rule created with EasyRule, it can no longer be opened and edited with EasyRule. When the User Code Entry Points are present, user code can be inserted between each pair of entry points using the ISPF editor and the rule can continue to be opened with EasyRule. If modifications are made outside of these entry points, EasyRule can no longer be used to open the rule.

Specify Conditions



```
EasyRule ---------------------------------------------------------------------
OPTION ===> D

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

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

5 - 16

Specify Conditions

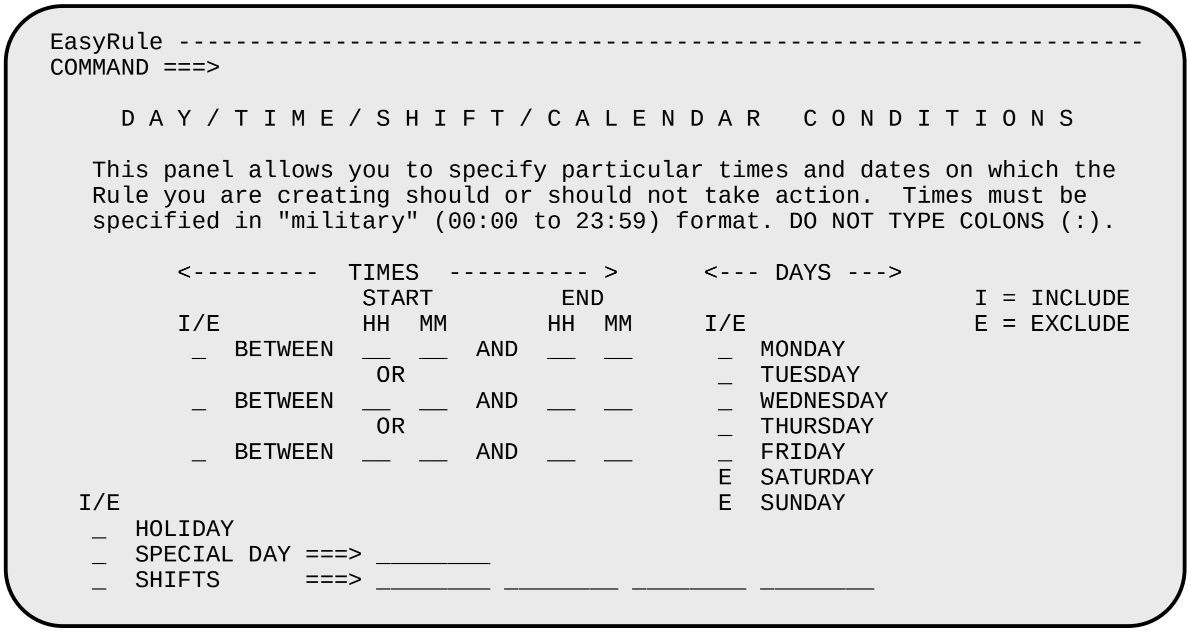
Now, we will use the Conditions Menu to choose conditions for EasyRule to use as selection criteria for the rule. These conditions are in addition to the primary selection criterion you specified on the Primary Event Specification panel. A unique Conditions Menu exists for each rule type. A sample panel, showing the Conditions Menu for message rules, appears above.

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.

Action: Type a D (Day/Time/Shift/Calendar) in the Option field and then press Enter. The Day/Time/Shift/Calendar Conditions panel displays.

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

Change Day



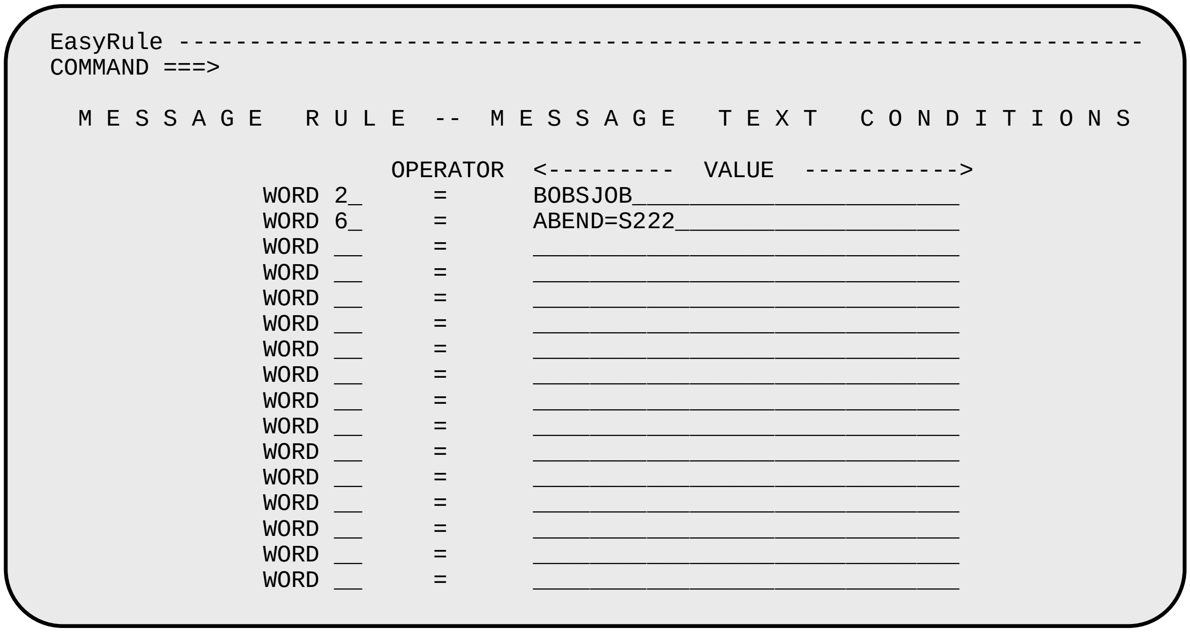
```
EasyRule --------------------------------------------------------------------
COMMAND ===>

     D A Y / T I M E / S H I F T / C A L E N D A R   C O N D I T I O N S

   This panel allows you to specify particular times and dates on which the
   Rule you are creating should or should not take action.  Times must be
   specified in "military" (00:00 to 23:59) format. DO NOT TYPE COLONS (:).

         <---------  TIMES  ---------- >      <--- DAYS --->
                      START         END                           I = INCLUDE
         I/E          HH  MM       HH  MM     I/E                 E = EXCLUDE
          _  BETWEEN  __  __  AND  __  __      _  MONDAY
                       OR                      _  TUESDAY
          _  BETWEEN  __  __  AND  __  __      _  WEDNESDAY
                       OR                      _  THURSDAY
          _  BETWEEN  __  __  AND  __  __      _  FRIDAY
                                               E  SATURDAY
  I/E                                          E  SUNDAY
   _  HOLIDAY
   _  SPECIAL DAY ===> ________
   _  SHIFTS      ===> ________ ________ ________ ________
```

Change Day

Next, we will use the Day/Time/Shift/Calendar Conditions panel, shown above, to specify conditions relating to particular days of the week.

Action: Type an E next to the SATURDAY and SUNDAY fields to exclude these days from the rule’s action and then press PF3 to return to the Conditions Menu.

Verify Message Wording



```
EasyRule --------------------------------------------------------------------
COMMAND ===>

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

                       OPERATOR  <---------  VALUE  ----------->
              WORD 2_     =      BOBSJOB_______________________
              WORD 6_     =      ABEND=S222____________________
              WORD __     =      ______________________________
              WORD __     =      ______________________________
              WORD __     =      ______________________________
              WORD __     =      ______________________________
              WORD __     =      ______________________________
              WORD __     =      ______________________________
              WORD __     =      ______________________________
              WORD __     =      ______________________________
              WORD __     =      ______________________________
              WORD __     =      ______________________________
              WORD __     =      ______________________________
              WORD __     =      ______________________________
              WORD __     =      ______________________________
              WORD __     =      ______________________________
```

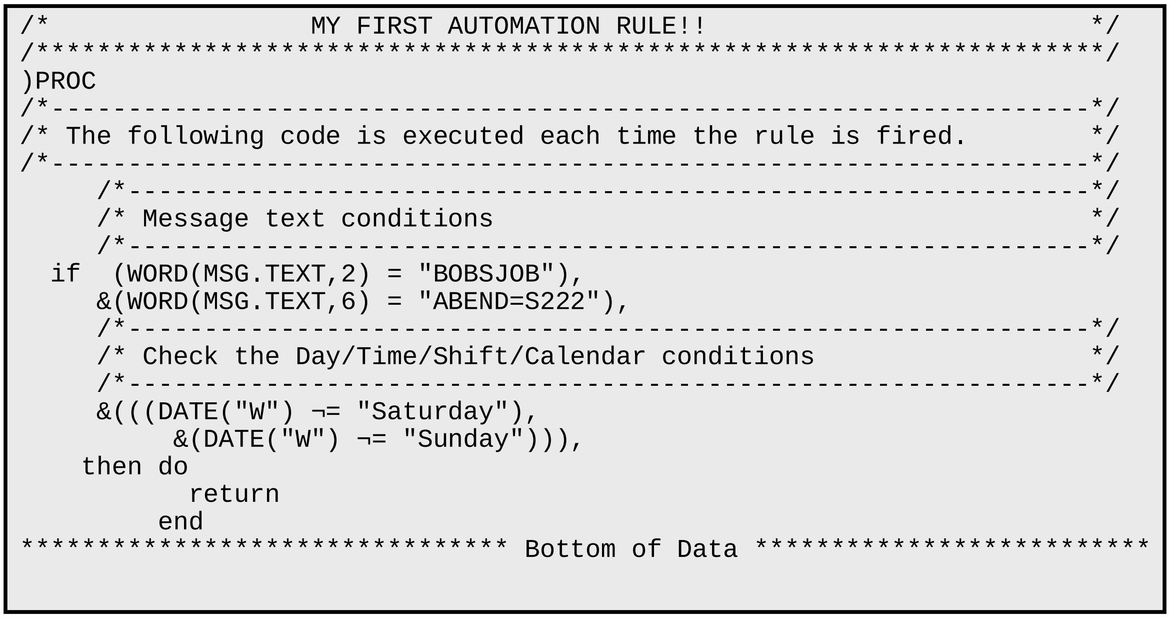
5 - 18

Verify Message Wording

Now, we will enter option 1 on the Conditions Menu to change the text of the message. The Message Text Conditions panel displays, an example of which is shown above.

Action: Complete the panel as shown above and then press PF3 until the EasyRule Final Options Menu appears.

Check Progress



```
/*                 MY FIRST AUTOMATION RULE!!                         */
/**********************************************************************/
)PROC
/*--------------------------------------------------------------------*/
/* The following code is executed each time the rule is fired.       */
/*--------------------------------------------------------------------*/
     /*---------------------------------------------------------------*/
     /* Message text conditions                                       */
     /*---------------------------------------------------------------*/
  if  (WORD(MSG.TEXT,2) = "BOBSJOB"),
     &(WORD(MSG.TEXT,6) = "ABEND=S222"),
     /*---------------------------------------------------------------*/
     /* Check the Day/Time/Shift/Calendar conditions                  */
     /*---------------------------------------------------------------*/
     &(((DATE("W") ¬= "Saturday"),
          &(DATE("W") ¬= "Sunday"))),
    then do
           return
         end
******************************** Bottom of Data *************************
```

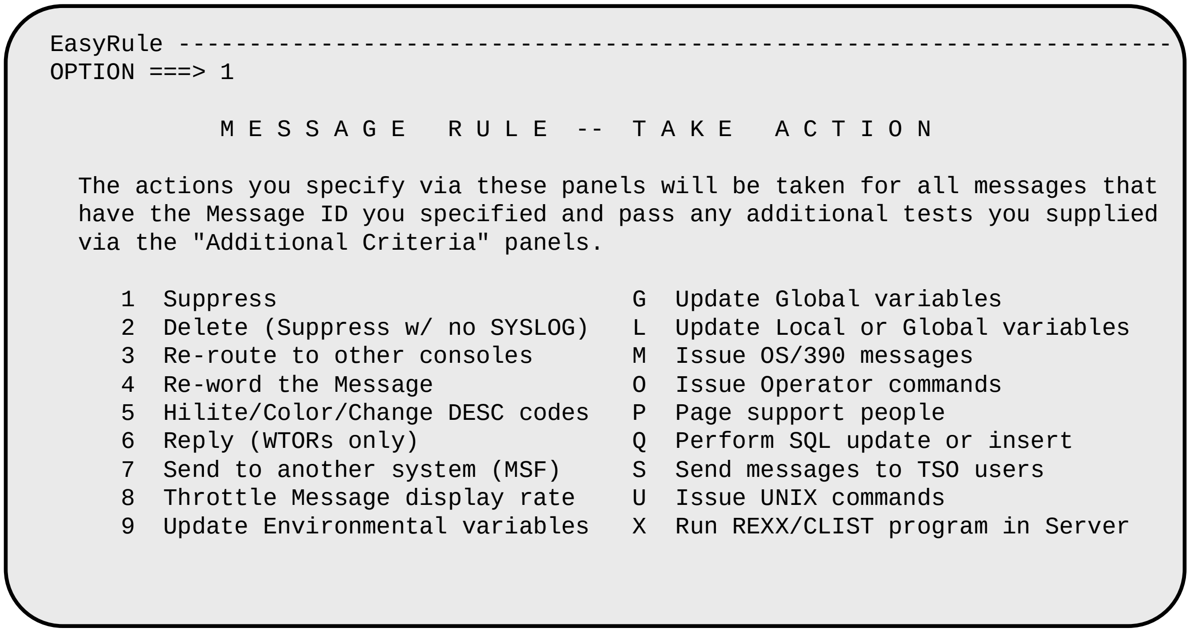
5 - 19

Check Progress

Once again, we will use the EasyRule Final Options Menu (option 3 - Browse) to view the progress of our rule.

Action: When you are finished viewing the code, press PF3. The EasyRule Final Options Menu appears. Enter a 4 (Alter) in the Option field and then press Enter to continue writing the rule.

Specify an Action



```
EasyRule ------------------------------------------------------------------
OPTION ===> 1

            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  Delete (Suppress w/ no SYSLOG)    L  Update Local or Global variables
     3  Re-route to other consoles        M  Issue OS/390 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
```

5 - 20

Specify an Action

Next, we will use the Take Action panel to specify an action for the rule to take when it is enabled. To access this panel, type a 4 in the Option field on the Message Rule Main Menu and then press Enter.

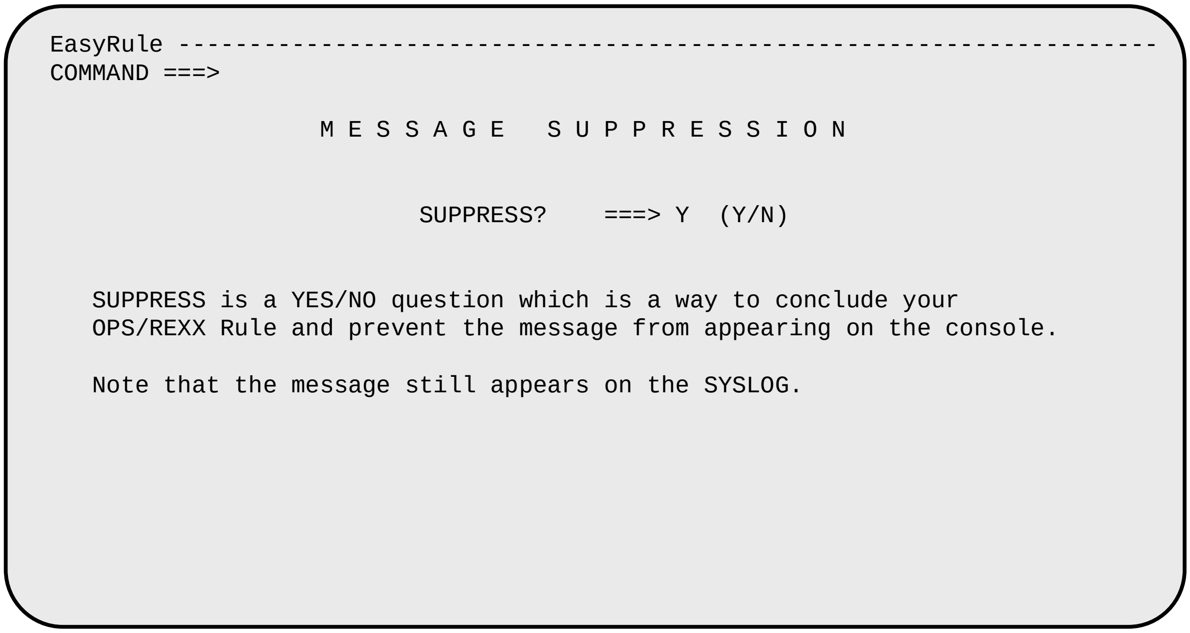
A unique Take Action panel exists for each rule type. The above sample display shows the Take Action panel for message rules:

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.

Action: Type a 1 (Suppress) in the Option field and then press Enter. The Message Suppression panel displays.

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

Specify an Action (continued)



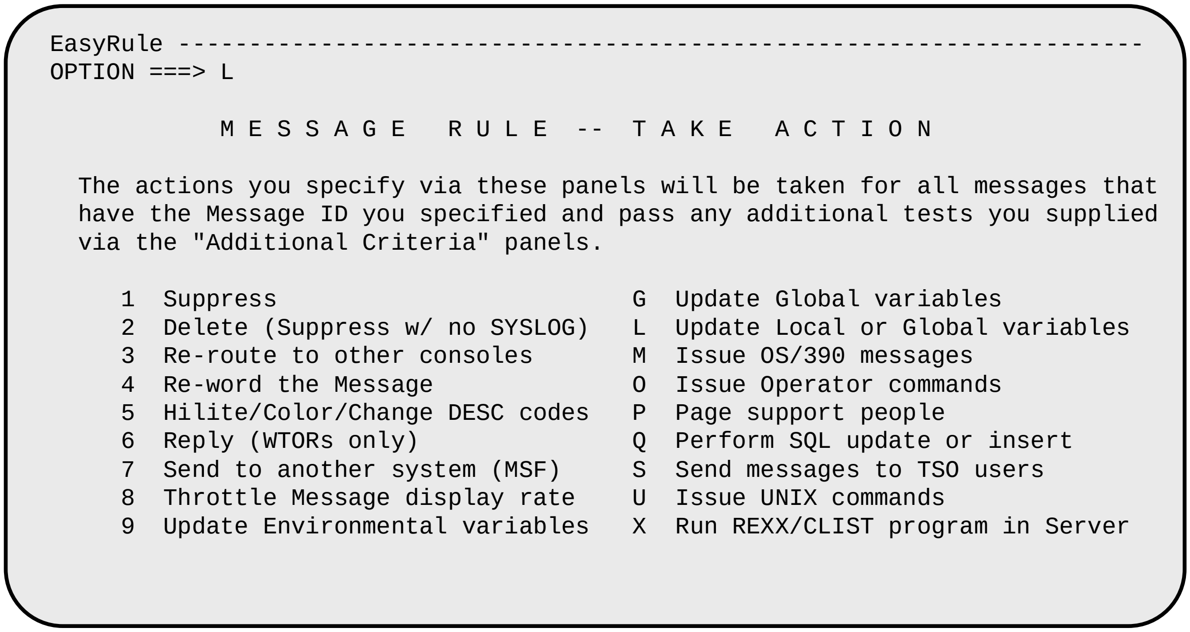
```
EasyRule -------------------------------------------------------------------
COMMAND ===>

                  M E S S A G E   S U P P R E S S I O N


                         SUPPRESS?    ===> Y  (Y/N)


   SUPPRESS is a YES/NO question which is a way to conclude your
   OPS/REXX Rule and prevent the message from appearing on the console.

   Note that the message still appears on the SYSLOG.
```

Specify an Action

We will use the Message Suppression panel, shown above, to prevent the message from appearing on the console.

Action: Type a Y in the SUPPRESS field and then press PF3 to return to the Take Action panel.

Specify Another Action



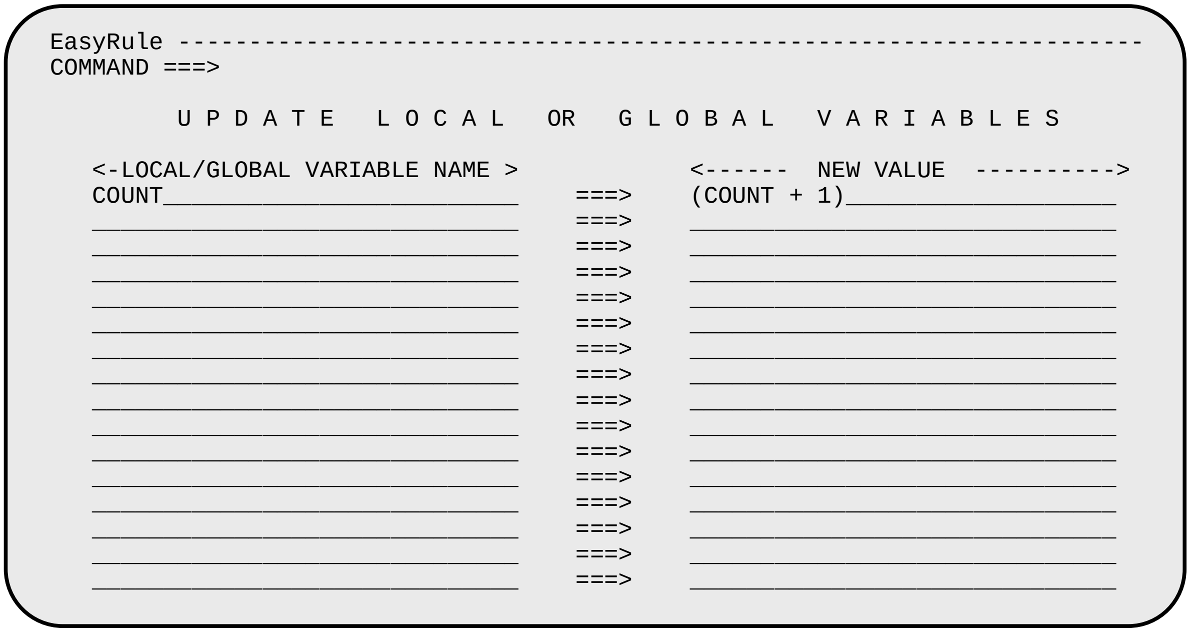
```
EasyRule -----------------------------------------------------------------
OPTION ===> L

            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  Delete (Suppress w/ no SYSLOG)    L  Update Local or Global variables
     3  Re-route to other consoles        M  Issue OS/390 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
```

Specify Another Action

Let’s specify another action for the rule using the Take Action panel.

Action: Enter an L in the Option field and then press Enter. The Update Local or Global Variables panel appears.

Track Occurrences



```
EasyRule --------------------------------------------------------------------
COMMAND ===>

         U P D A T E   L O C A L   OR   G L O B A L   V A R I A B L E S

   <-LOCAL/GLOBAL VARIABLE NAME >           <------  NEW VALUE  ---------->
   COUNT_________________________    ===>   (COUNT + 1)___________________
   ______________________________    ===>   ______________________________
   ______________________________    ===>   ______________________________
   ______________________________    ===>   ______________________________
   ______________________________    ===>   ______________________________
   ______________________________    ===>   ______________________________
   ______________________________    ===>   ______________________________
   ______________________________    ===>   ______________________________
   ______________________________    ===>   ______________________________
   ______________________________    ===>   ______________________________
   ______________________________    ===>   ______________________________
   ______________________________    ===>   ______________________________
   ______________________________    ===>   ______________________________
   ______________________________    ===>   ______________________________
   ______________________________    ===>   ______________________________
   ______________________________    ===>   ______________________________
```

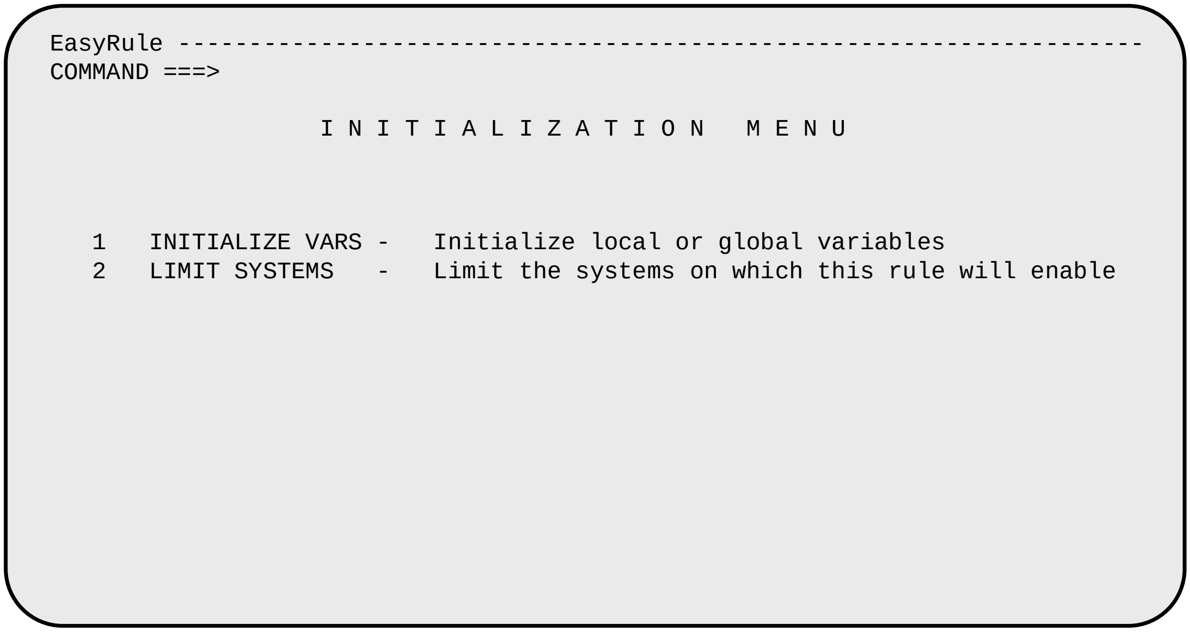
5 - 23

Track Occurrences

Now, we will assign a value to a local variable that will keep track of the number of times the rule executes.

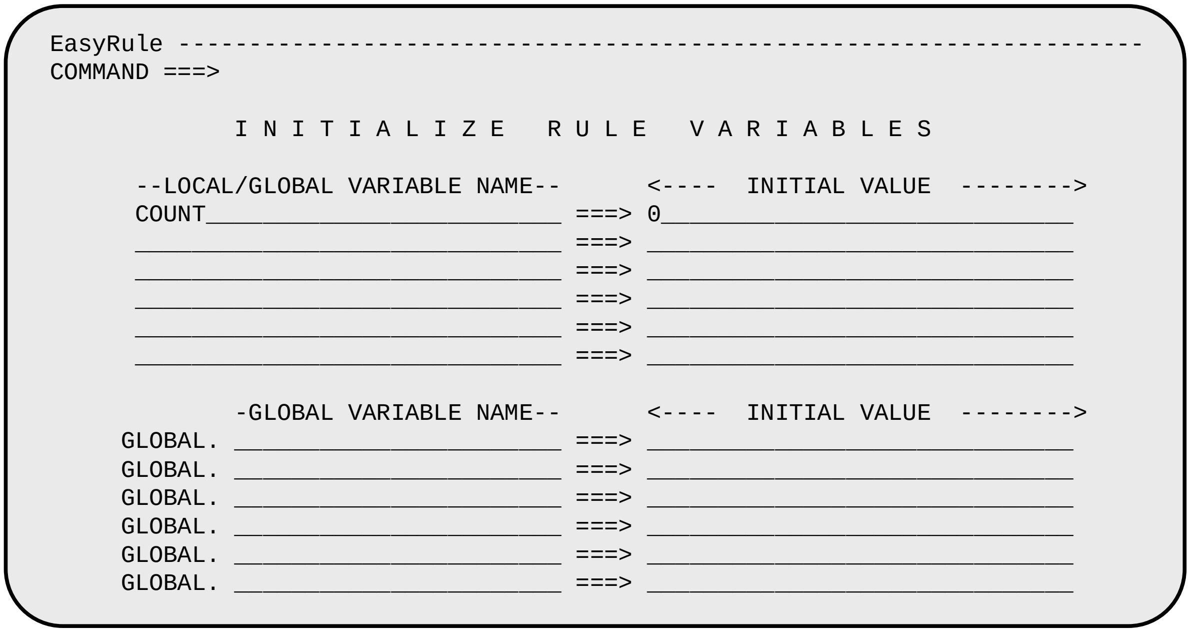
Action: Complete the panel as shown above and then press PF3 until the Message Rule Main Menu appears.

Initialize



```
EasyRule -----------------------------------------------------------------------
COMMAND ===>

                   I N I T I A L I Z A T I O N   M E N U



   1   INITIALIZE VARS -   Initialize local or global variables
   2   LIMIT SYSTEMS   -   Limit the systems on which this rule will enable
```

Initialize Variables



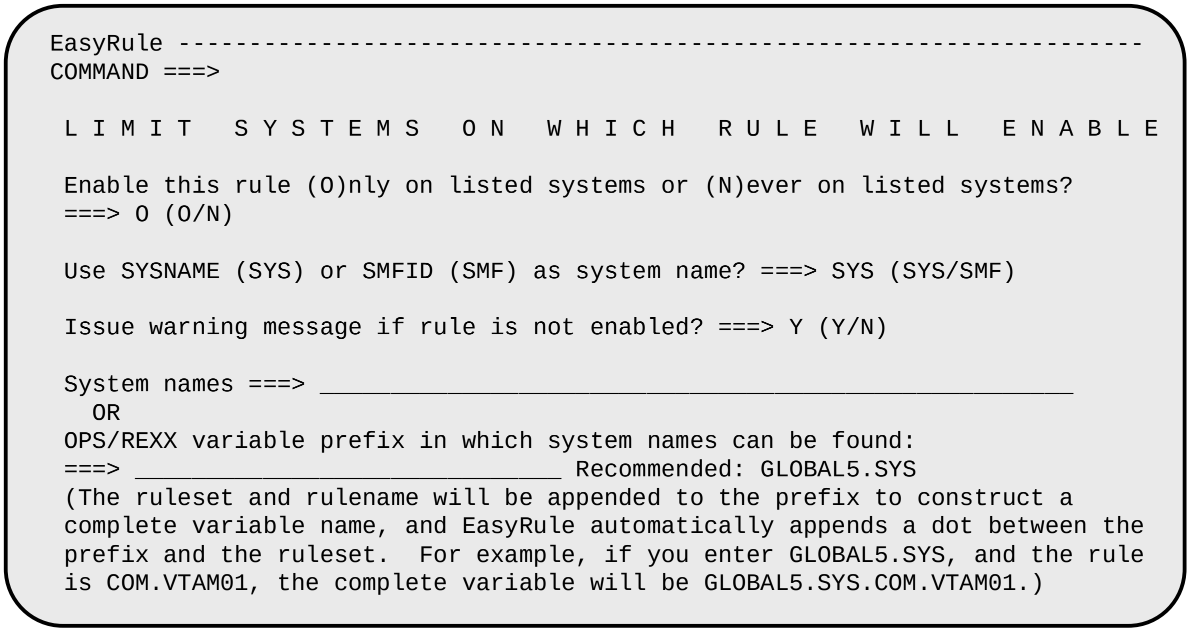
```
EasyRule -------------------------------------------------------------------
COMMAND ===>

             I N I T I A L I Z E   R U L E   V A R I A B L E S

      --LOCAL/GLOBAL VARIABLE NAME--     <----  INITIAL VALUE  -------->
      COUNT________________________ ===> 0______________________________
      _____________________________ ===> _______________________________
      _____________________________ ===> _______________________________
      _____________________________ ===> _______________________________
      _____________________________ ===> _______________________________
      _____________________________ ===> _______________________________

             -GLOBAL VARIABLE NAME--     <----  INITIAL VALUE  -------->
     GLOBAL. ______________________ ===> _______________________________
     GLOBAL. ______________________ ===> _______________________________
     GLOBAL. ______________________ ===> _______________________________
     GLOBAL. ______________________ ===> _______________________________
     GLOBAL. ______________________ ===> _______________________________
     GLOBAL. ______________________ ===> _______________________________
```

Initialize Variables

Next, we will use the Initialize Rule Variables panel, shown below, to initialize the value of the variable. You use this panel to set variables to specific values before the rule executes for the first time. To access this panel, type a 5 in the Option field on the Message Rule Main Menu and then press Enter.

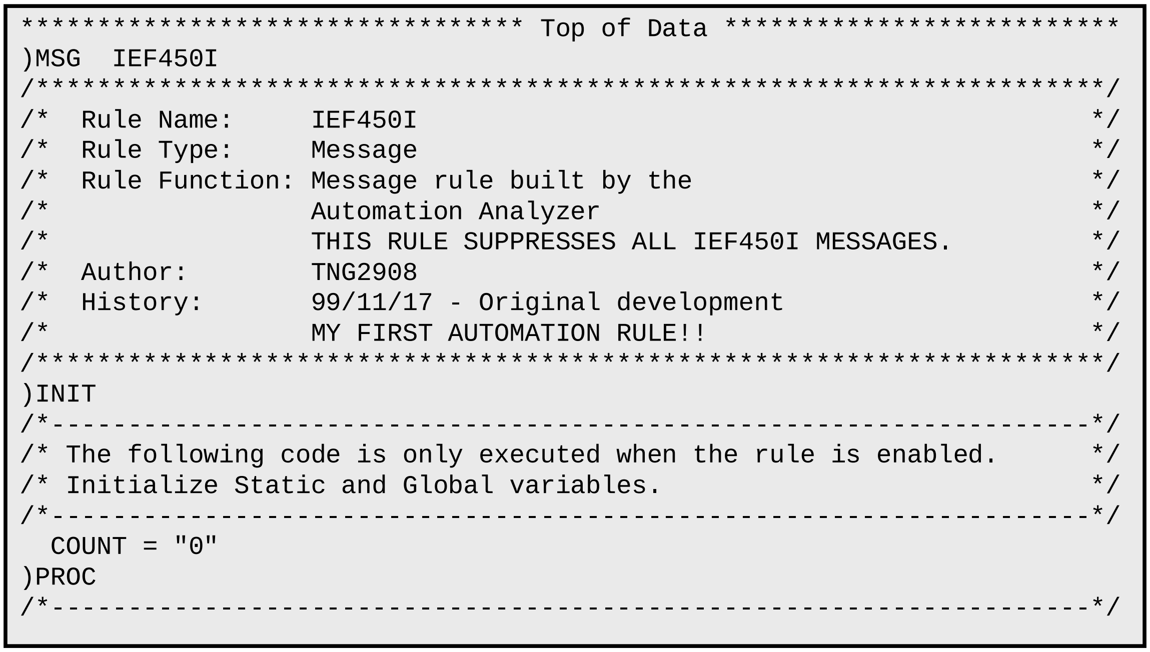
Action: Complete the panel as shown above and then press PF3. The Message Rule Main Menu displays.

Limit Systems



```
EasyRule ---------------------------------------------------------------------
COMMAND ===>

 L I M I T   S Y S T E M S   O N   W H I C H   R U L E   W I L L   E N A B L E

 Enable this rule (O)nly on listed systems or (N)ever on listed systems?
 ===> O (O/N)

 Use SYSNAME (SYS) or SMFID (SMF) as system name? ===> SYS (SYS/SMF)

 Issue warning message if rule is not enabled? ===> Y (Y/N)

 System names ===> _____________________________________________________
   OR
 OPS/REXX variable prefix in which system names can be found:
 ===> ______________________________ Recommended: GLOBAL5.SYS
 (The ruleset and rulename will be appended to the prefix to construct a
 complete variable name, and EasyRule automatically appends a dot between the
 prefix and the ruleset.  For example, if you enter GLOBAL5.SYS, and the rule
 is COM.VTAM01, the complete variable will be GLOBAL5.SYS.COM.VTAM01.)
```

Is Rule Complete?

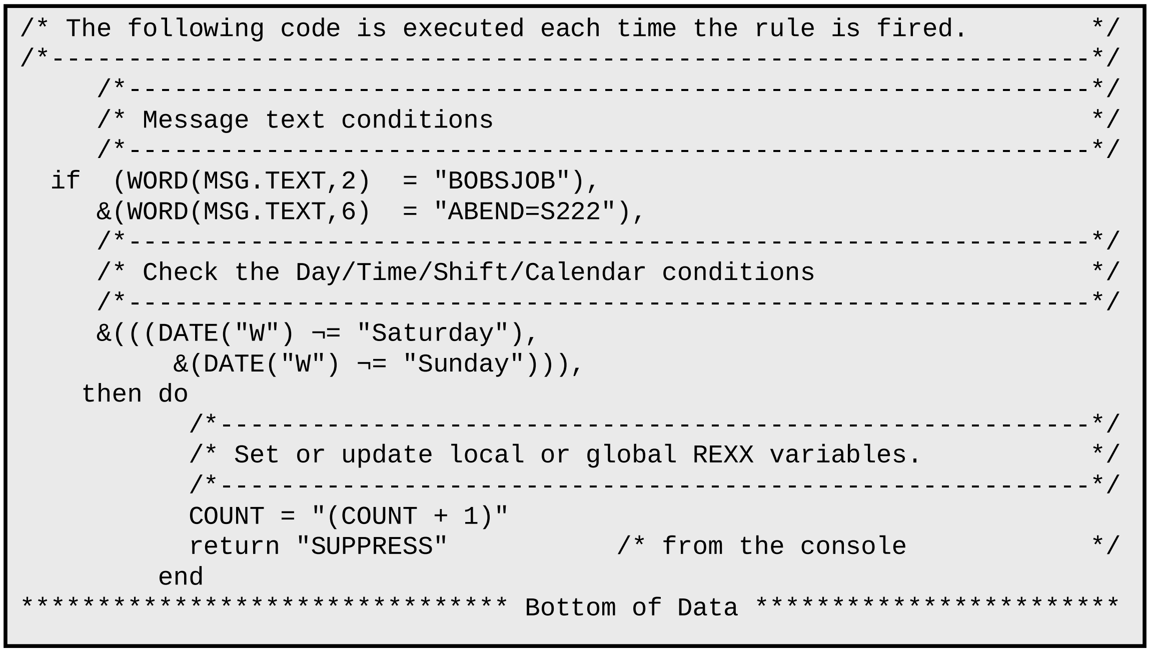


```
********************************* Top of Data **************************
)MSG  IEF450I
/**********************************************************************/
/*  Rule Name:     IEF450I                                            */
/*  Rule Type:     Message                                            */
/*  Rule Function: Message rule built by the                          */
/*                 Automation Analyzer                                */
/*                 THIS RULE SUPPRESSES ALL IEF450I MESSAGES.         */
/*  Author:        TNG2908                                            */
/*  History:       99/11/17 - Original development                    */
/*                 MY FIRST AUTOMATION RULE!!                         */
/**********************************************************************/
)INIT
/*--------------------------------------------------------------------*/
/* The following code is only executed when the rule is enabled.      */
/* Initialize Static and Global variables.                            */
/*--------------------------------------------------------------------*/
  COUNT = "0"
)PROC
/*--------------------------------------------------------------------*/
```

Is Rule Complete?

Once again, we will use the EasyRule Final Options Menu to view our rule. To do so, type a 3 (Browse) in the Option field and then press Enter.

Is Rule Complete? (continued)



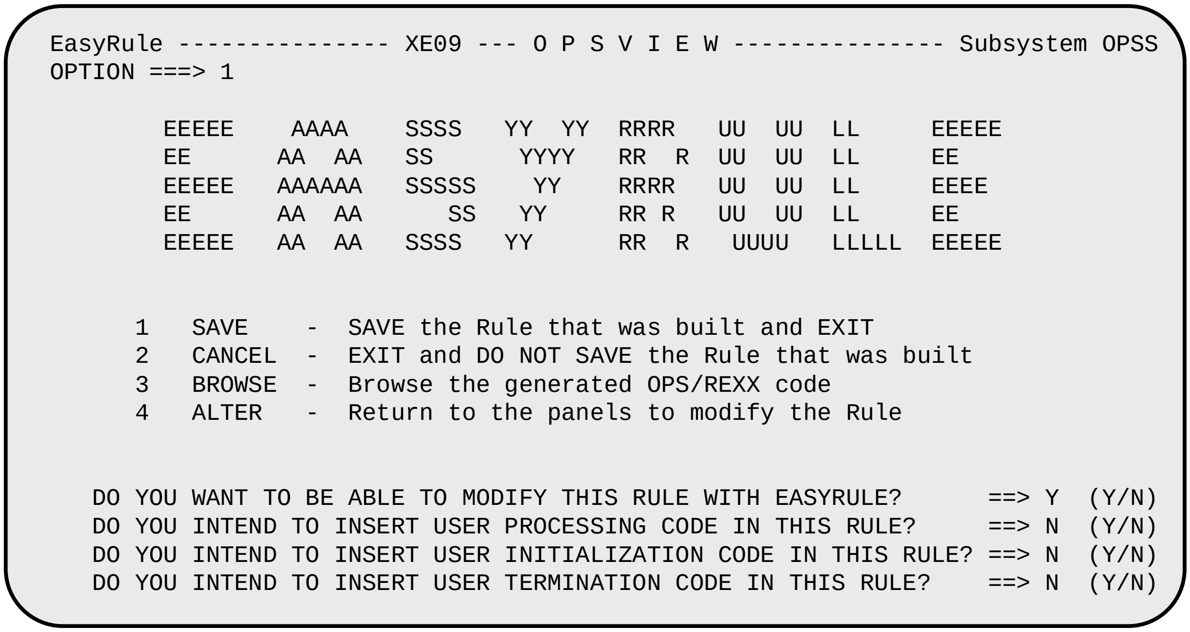
```
/* The following code is executed each time the rule is fired.       */
/*--------------------------------------------------------------------*/
     /*---------------------------------------------------------------*/
     /* Message text conditions                                       */
     /*---------------------------------------------------------------*/
  if  (WORD(MSG.TEXT,2)  = "BOBSJOB"),
     &(WORD(MSG.TEXT,6)  = "ABEND=S222"),
     /*---------------------------------------------------------------*/
     /* Check the Day/Time/Shift/Calendar conditions                  */
     /*---------------------------------------------------------------*/
     &(((DATE("W") ¬= "Saturday"),
          &(DATE("W") ¬= "Sunday"))),
    then do
           /*---------------------------------------------------------*/
           /* Set or update local or global REXX variables.           */
           /*---------------------------------------------------------*/
           COUNT = "(COUNT + 1)"
           return "SUPPRESS"           /* from the console            */
         end
******************************** Bottom of Data ************************
```

5 - 28

Is Rule Complete?

Action: When you are finished viewing the code, press PF3 until the EasyRule Final Options Menu appears.

Save



```
EasyRule --------------- XE09 --- O P S V I E W --------------- Subsystem OPSS
OPTION ===> 1

        EEEEE    AAAA    SSSS   YY  YY  RRRR   UU  UU  LL     EEEEE
        EE      AA  AA   SS       YYYY   RR  R  UU  UU  LL     EE
        EEEEE   AAAAAA   SSSSS    YY    RRRR   UU  UU  LL     EEEE
        EE      AA  AA      SS   YY     RR R   UU  UU  LL     EE
        EEEEE   AA  AA   SSSS   YY      RR  R   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)
```

Save

If you are pleased with your rule, the last step is to save it. To do so, type a 1 (Save) in the Option field on the EasyRule Final Options Menu and then press Enter.

More About the EasyRule Final Options Menu

The following options are available on the EasyRule Final Options Menu:

1. Save

   EasyRule saves the rule into the member you specified on the EasyRule Primary Panel. (If you prefer, you can press PF3 to achieve the same result.)

2. Cancel

   EasyRule ignores all of your panel entries. No rule is created or updated. (If you prefer, you can enter the CANCEL command into the panel's Option field to achieve the same result.)

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).

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 rule's specifications 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.

Making 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.

EasyRule Help

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.

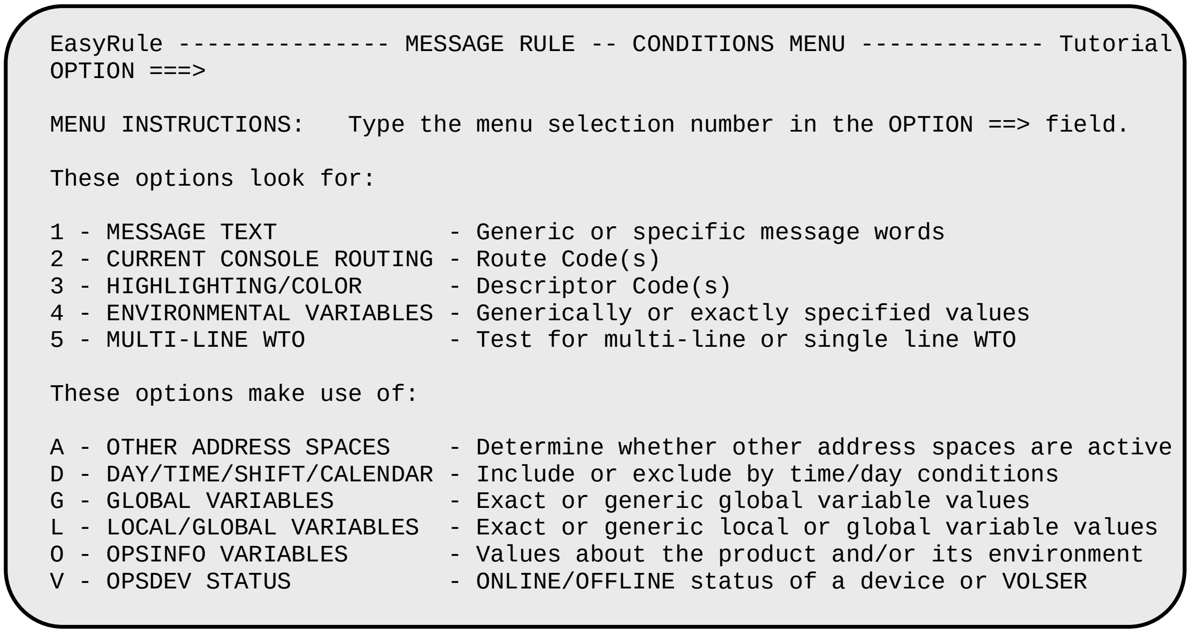
Accessing EasyRule Help

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

The Four Basic Types of EasyRule Help Panels

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

EasyRule Menu Help



```
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
```

5 - 32

EasyRule Menu Help

If you access help from an EasyRule menu panel, you access the help panel for that menu. For example, if you press PF1/13 from the Conditions Menu for message rules, the above menu help panel appears.

How Menu Help Panels Present Information:

- Menu help panels provide a little more detail about each of the available menu options.

Actions You May Take on a Menu Help Panel:

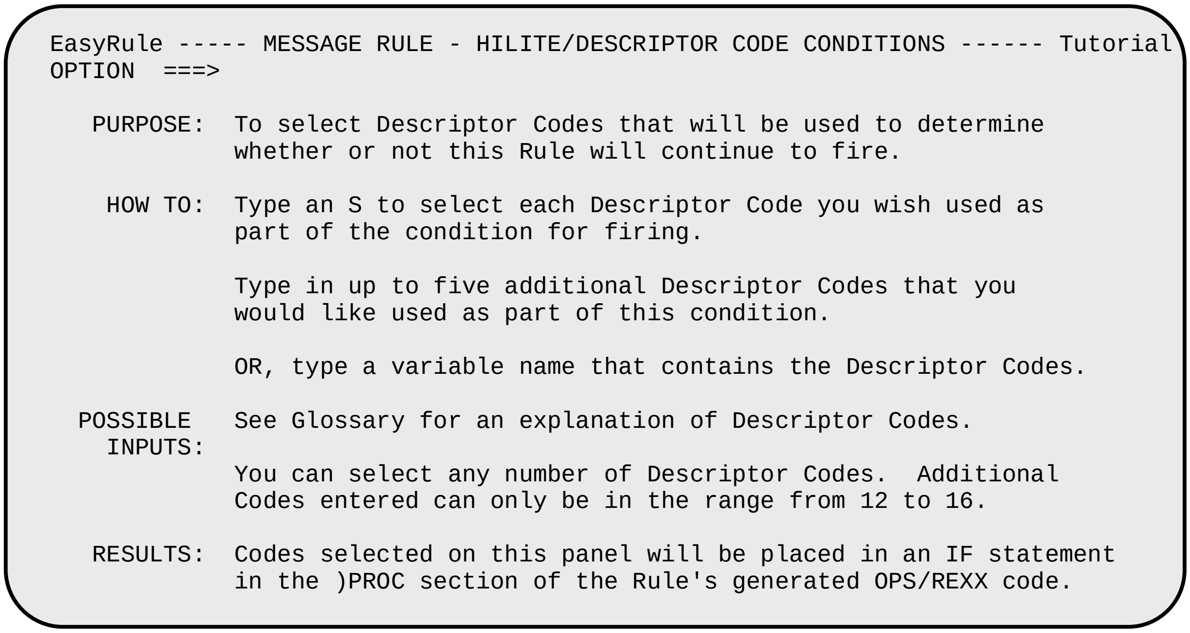
- Enter a selection number or letter in Option field.

  A more detailed help panel for the selected option appears.

- PF3

  Your EasyRule help session is terminated.

EasyRule Standard Help



```
EasyRule ----- MESSAGE RULE - HILITE/DESCRIPTOR CODE CONDITIONS ------ Tutorial
OPTION  ===>

   PURPOSE:  To select Descriptor Codes that will be used to determine
             whether or not this Rule will continue to fire.

    HOW TO:  Type an S to select each Descriptor Code you wish used as
             part of the condition for firing.

             Type in up to five additional Descriptor Codes that you
             would like used as part of this condition.

             OR, type a variable name that contains the Descriptor Codes.

  POSSIBLE   See Glossary for an explanation of Descriptor Codes.
    INPUTS:
             You can select any number of Descriptor Codes.  Additional
             Codes entered can only be in the range from 12 to 16.

   RESULTS:  Codes selected on this panel will be placed in an IF statement
             in the )PROC section of the Rule's generated OPS/REXX code.
```

EasyRule Standard Help

Another type of help panel you can access from EasyRule is a standard help panel. For example, if you press PF1/13 from the Message Rule - Descriptor Code Conditions panel, the above standard help panel appears.

How Standard Help Panels Present Information

- 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.

You may take these actions on most standard help panels:

- Enter E in Option field.

  A help example panel appears. It will present 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.

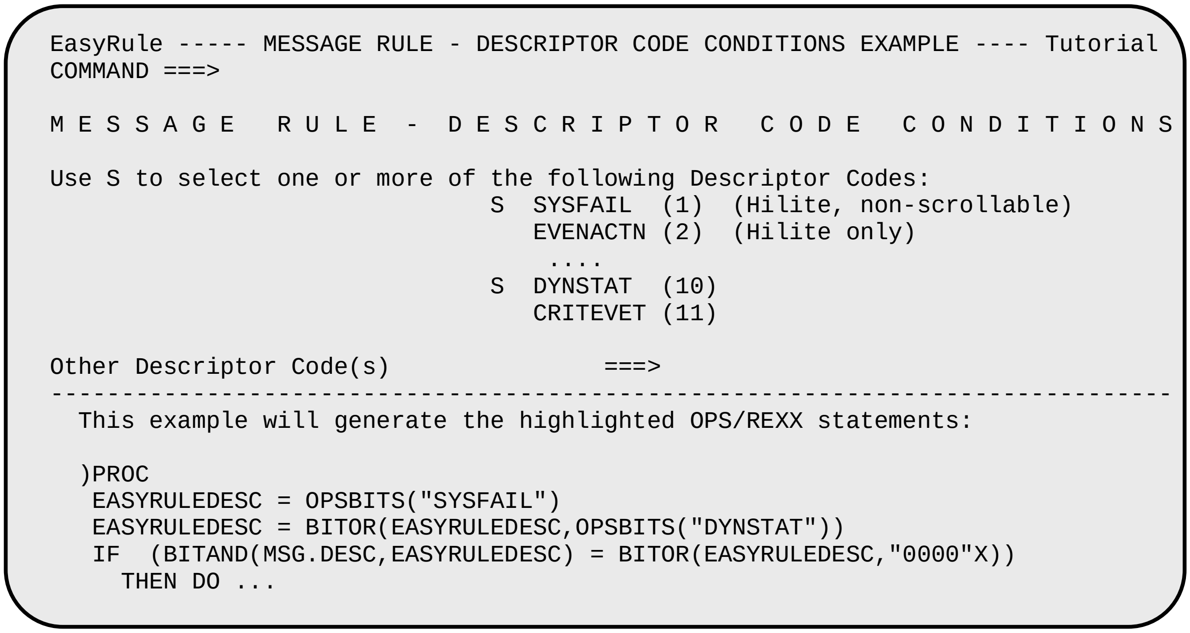
- Press Enter.

  You are presented with a second standard help panel. (This applies to two-part standard help panels only.)

- Press PF3.

  Your EasyRule help session is terminated.

EasyRule Example Help



```
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 ...
```

EasyRule Example Help

On most of EasyRule's standard help panels, you can enter E in the Option field to see an example of a correctly filled-in panel, along with the OPS/REXX code that EasyRule would generate from those entries. For example, if you enter E in the Option field of the panel shown below, the above help example panel appears.

How Help Example Panels Present Information

- The top half of the panel shows the EasyRule panel that you wanted an example of. The panel is filled in with a typical set of entries, which are highlighted. (Sometimes, due to space limitations, not all panel lines appear.)
- 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. (Sometimes, due to space limitations, not all lines of generated code appear.)

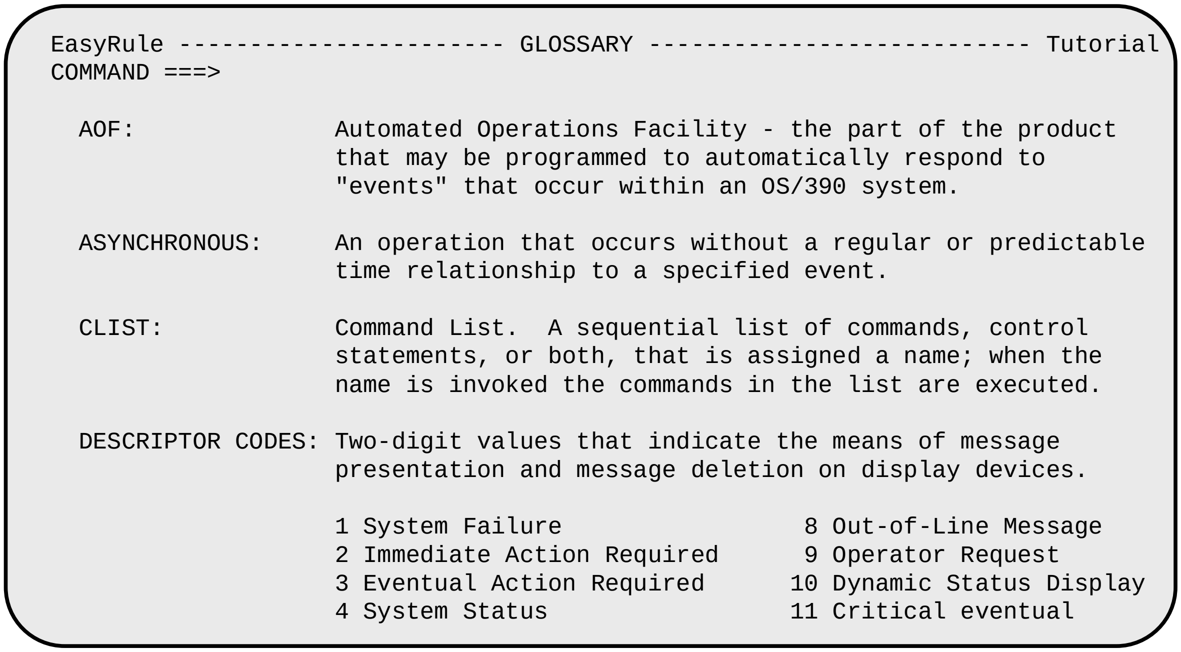
Actions You May Take on a Help Example Panel

- Press Enter.

  You are returned to the previous help panel.
- Press PF3.

  Your EasyRule help session is terminated.

EasyRule Glossary



```
EasyRule ---------------------- GLOSSARY -------------------------- Tutorial
COMMAND ===>

  AOF:              Automated Operations Facility - the part of the product
                    that may be programmed to automatically respond to
                    "events" that occur within an OS/390 system.

  ASYNCHRONOUS:     An operation that occurs without a regular or predictable
                    time relationship to a specified event.

  CLIST:            Command List.  A sequential list of commands, control
                    statements, or both, that is assigned a name; when the
                    name is invoked the commands in the list are executed.

  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
```

EasyRule Glossary

On most of EasyRule’s standard help panels, you can enter G in the Option field to access EasyRule’s glossary. The fabove sample panel shows a help glossary.

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

Actions You May Take on a Help Glossary Panel:

- Enter B in Command field.

  You are moved backward in the glossary.

- Enter F in Command field.

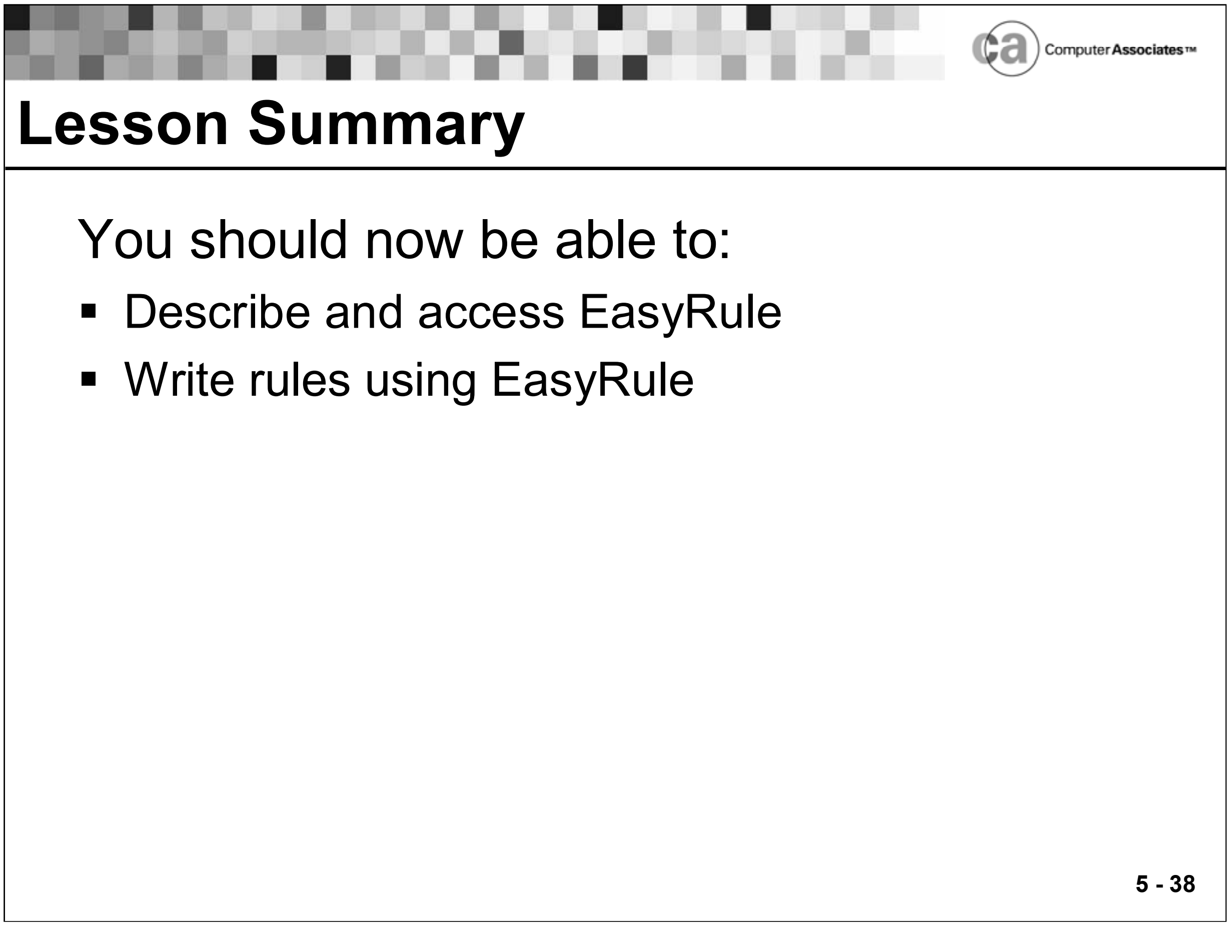
  You are moved forward in the glossary.

- Press Enter.

  You are returned to the previous help panel.

- Press PF3.

  Your EasyRule help session is terminated.



Lesson Summary

You should now be able to:

- Describe and access EasyRule
- Write rules using EasyRule

5 - 38

Lesson 5 Activity

5 - 39

Lesson 5 Activity – Solving a Problem With EasyRule

Scenario

- You need to be aware of NOT CATALOGED 2 conditions because they are indicative of production problems. These conditions are identified in IEF287I messages.
- IEF285I messages are related normal messages that do not require action.
- You want to increase the visibility of the NOT CATALOGED 2 conditions. The presence of IEF285I messages creates visual noise, making it difficult for you to see and respond to the important IEF287I messages.

Task

- Write two rules.

  The first rule, called MNSTATUS, should not only suppress IEF285I messages from displaying on the console, but also keep them from appearing in the SYSLOG.

Lesson 5 Activity (continued)

The second rule, called NOTCTLG, should highlight IEF287I messages and post messages as described to the job log.

Steps to take:

1. Access the EasyRule Primary panel.
2. Specify a data set and member for the first rule:

   CLCS.<userid>.OPSRULES(MNSTATUS)
3. Select the type of rule you want to create.
4. Select MESSAGE ID from the Message Rule Main Menu.
5. Supply the primary selection criteria. (Hint: Specify a D in the Just Delete field.)
6. Select DOCUMENTATION from the Message Rule Main Menu.
7. Document the MNSTATUS rule.
8. Access the EasyRule Final Option Menu.
9. Review the OPS/REXX code that EasyRule built.
10. Save the MNSTATUS rule.
11. Specify a data set and member for the second rule:

    CLCS.<userid>.OPSRULES(NOTCTLG)
12. Select the type of rule you want to create.
13. Select MESSAGE ID from the Message Rule Main Menu.
14. Supply the primary selection criteria.
15. Select DOCUMENTATION from the Message Rule Main Menu.
16. Document the rule you are creating.
17. Select 5 from the Take Action menu.
18. Specify the descriptor code as SYSFAIL.
19. Select ACTIONS from the Message Rule Main Menu.
20. Select O from the Take Action menu.
21. Write messages to the job log.

    Hint 1: The command to write messages to the Job Log is:

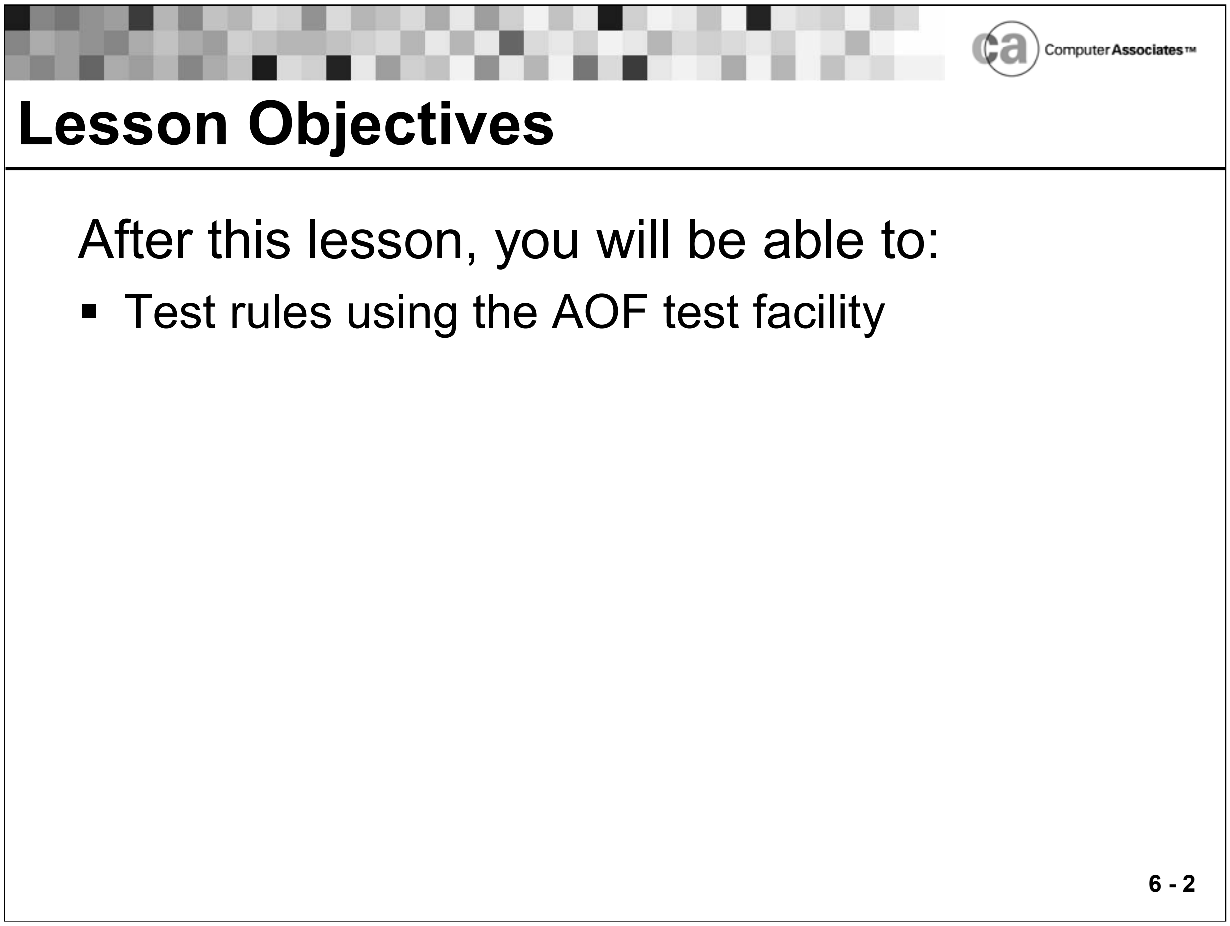
    $d m *jobnumber, msgtext*

    Hint 2: You must enclose variables in {} if they are specified in an Option O command string.
22. Access the EasyRule Final Options Menu.
23. Review the OPS/REXX code that EasyRule built.
24. Save the NOTCTLG rule.

Testing Rules

Lesson 6

ca.com

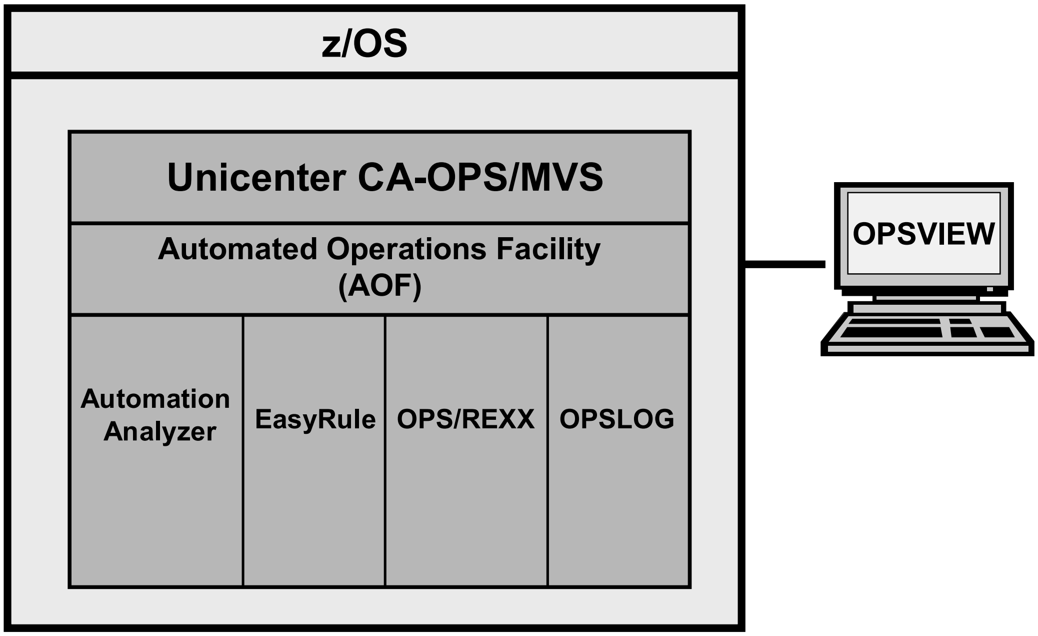


Lesson Objectives

After this lesson, you will be able to:

- Test rules using the AOF test facility

AOF Test Facility



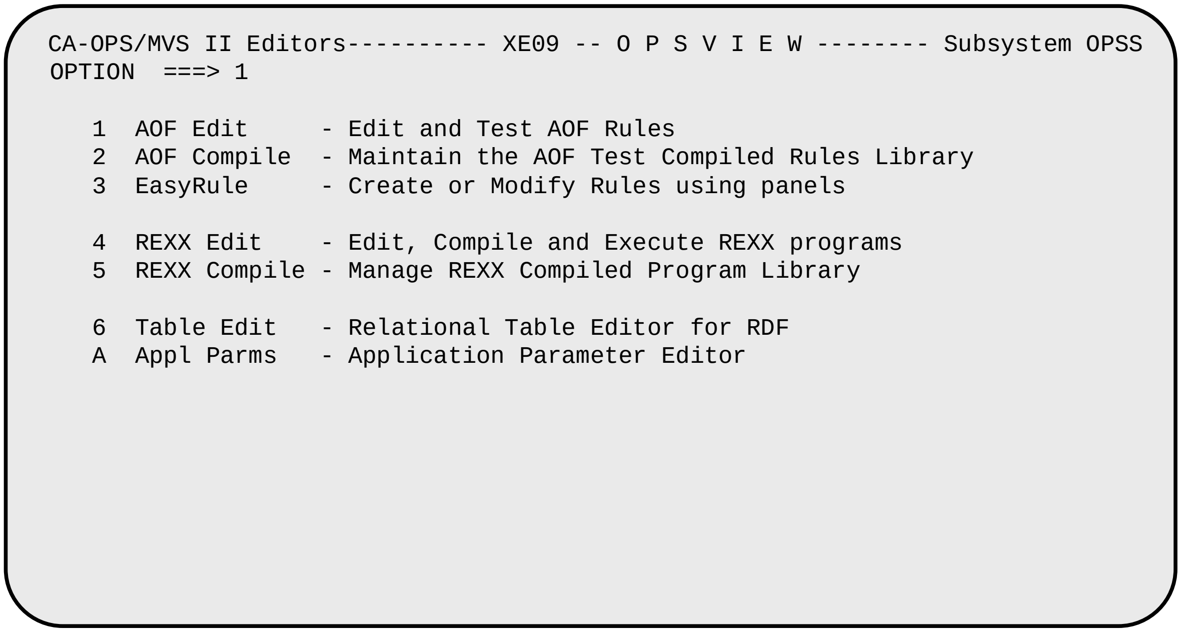
6 - 3

AOF Test Facility

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

The test facility lets you develop and test automation rules offline, before putting them into production. The primary components of the test facility are OPSVIEW option 2.1 (for editing and testing AOF rules) and 2.2 (for maintaining the AOF test compiled rules library).

OPSVIEW Editors



```
CA-OPS/MVS II Editors---------- XE09 -- O P S V I E W -------- Subsystem OPSS
OPTION  ===> 1

   1  AOF Edit     - Edit and Test AOF Rules
   2  AOF Compile  - Maintain the AOF Test Compiled Rules Library
   3  EasyRule     - Create or Modify Rules using panels

   4  REXX Edit    - Edit, Compile and Execute REXX programs
   5  REXX Compile - Manage REXX Compiled Program Library

   6  Table Edit   - Relational Table Editor for RDF
   A  Appl Parms   - Application Parameter Editor
```

6 - 4

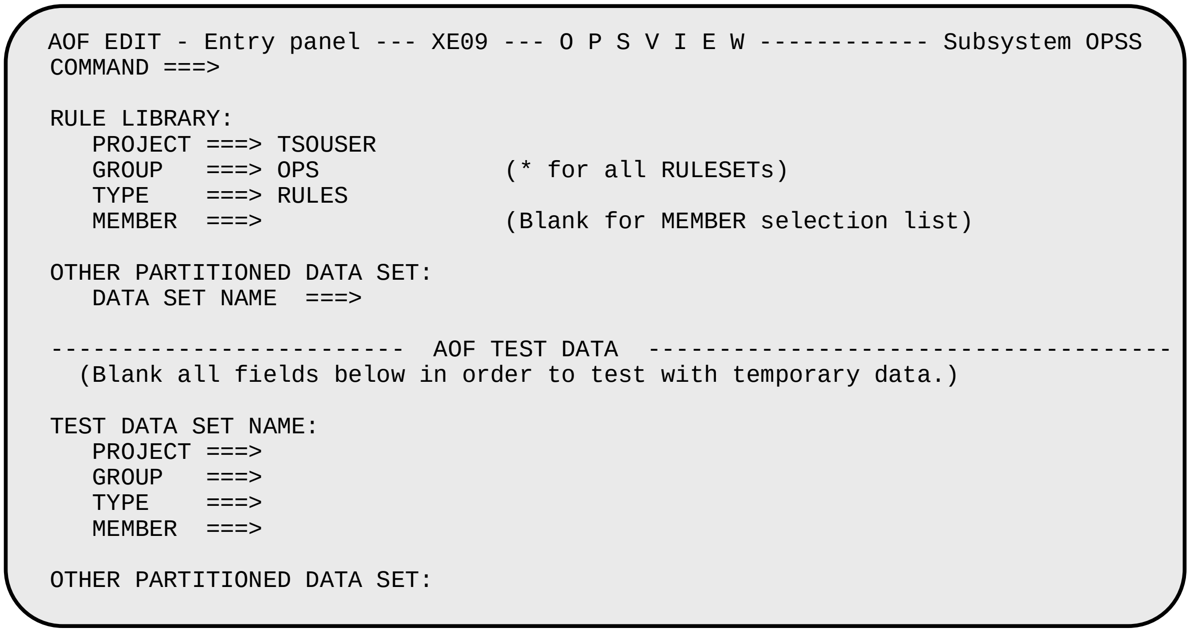
OPSVIEW Editors

To access the OPSVIEW Editors option, type a 2 on the OPSVIEW Primary Options Menu and then press Enter. You see a display similar to the one shown above. This panel is called the Editors menu.

Note: You can also access the Editor menu via the ISPF jump function. To do so, enter =2.1 into any valid field within OPSVIEW.

Action: To test rules, type a 1 (AOF Edit) in the Option field and then press Enter. The Specification Display panel appears.

AOF Edit



```
AOF EDIT - Entry panel --- XE09 --- O P S V I E W ----------- Subsystem OPSS
COMMAND ===>

RULE LIBRARY:
   PROJECT ===> TSOUSER
   GROUP   ===> OPS              (* for all RULESETs)
   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:
```

AOF Edit

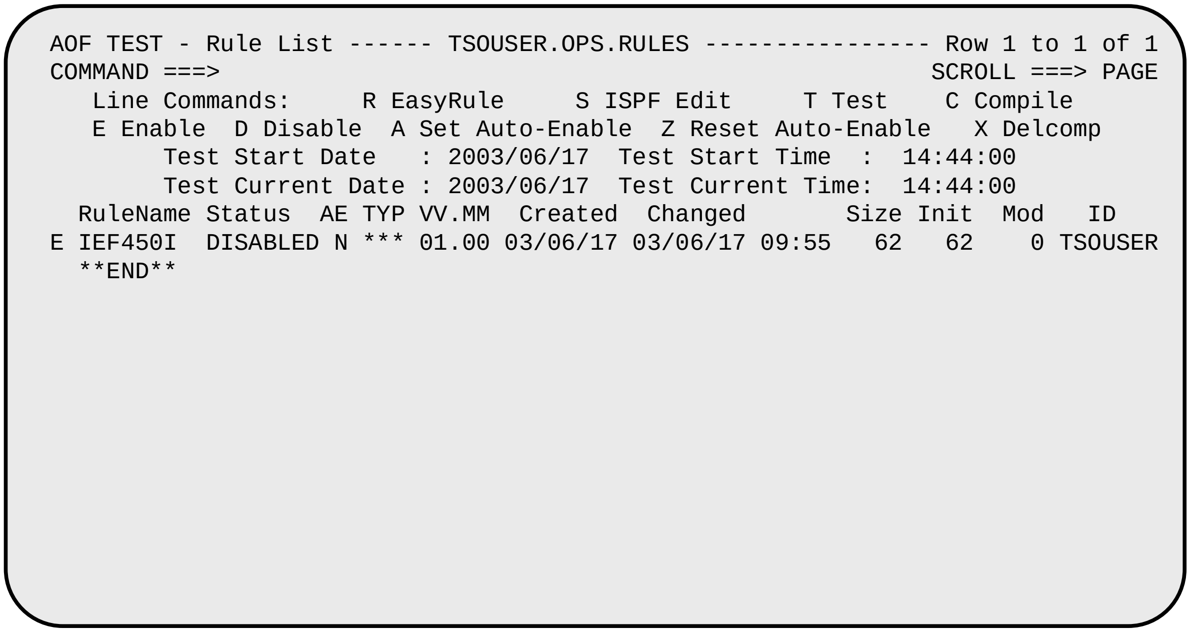
When you access OPSVIEW option 2.1 (AOF Edit), you see a display similar to the one shown above. This panel is called the Specification Display panel.

Action: Complete the panel as shown above and then press Enter. (In place of TSOUSER, type your user ID.) The Rule List panel appears.

There are two ways that you can display a list of rules:

- On the Specification Display panel, use the Project, Group, and Type fields to enter the name of an existing rule set, but leave the Member field blank.
- On the Rule Set List panel, issue the S line command.

Enable Rule



```
AOF TEST - Rule List ------ TSOUSER.OPS.RULES ---------------- Row 1 to 1 of 1
COMMAND ===>                                                 SCROLL ===> PAGE
   Line Commands:     R EasyRule     S ISPF Edit     T Test    C Compile
   E Enable  D Disable  A Set Auto-Enable  Z Reset Auto-Enable   X Delcomp
        Test Start Date   : 2003/06/17  Test Start Time  :  14:44:00
        Test Current Date : 2003/06/17  Test Current Time:  14:44:00
  RuleName Status  AE TYP VV.MM  Created  Changed       Size Init  Mod   ID
E IEF450I  DISABLED N *** 01.00 03/06/17 03/06/17 09:55   62   62    0 TSOUSER
  **END**
```

Enable Rule

The Rule List panel shows a listing of all the rules within a rule set.

You must enable a rule before you can test it. You can enable or disable rule sets, and rules within each rule set, independently of each other.

Action: To enable a rule, type an E (Enable) to the left of the desired rule name and then press Enter.

Two types of fields appear on the Rule List panel. The test date and test time fields that appear toward the top of the panel are modifiable. The remaining fields provide information about the individual rules in the rule set and cannot be modified.

Modifiable Fields on the Rule List Panel

- Test Start Date - The date you want the test to begin.
- Test Start Time - The time you want the test to begin.
- Test Current Date - The current date. May differ from Test Start Date.
- Test Current Time - The current time. May differ from Test Start Time.

Non-modifiable Fields on the Rule List Panel

- Rulename -Name of rule. Member name belonging to PDS, or rule set, that is named at top of panel. In sample panel, rules listed on the panel belong to the rule set named TSOUSER.OPS.RULES.)
- Status -Indicates whether the rule is enabled or disabled. You must use the E line command to enable a rule if you want to test it.
- AE -The rule set's auto-enable status.
- TYP -Type of rule. If Status is DISABLED, value of TYP field is ***. If Status is ENABLED, value of TYP field can be any of these values:

  ARM – Automatic Restart Management rule

  API – Application Program Interface

  CMD – Command rule

  DOM – Delete-operator-message rule

  EOJ – End-of-job rule

  EOM – End-of-memory rule

  EOS – End-of-step rule

  GLV – Global variable rule

  MSG – Message rule

  OMG – OMEGAMON rule

  REQ – Request rule

  SCR – Screen rule

  SEC – Security rule

  TLM – Time limit excession rule

  TOD – Time-of-day rule

  USS – UNIX System Services rule
- VV.MM -The version number and modification number of the rule.
- Created - The rule's creation date.
- Changed -The date and time of the last modification made to the rule.
- Size - The current number of lines in the rule.
- Init - The number of lines in this rule when it was first created.
- Mod - The number of lines in this rule that have been modified.
- ID - The TSO user ID of the last user who modified this rule.

Primary Commands for the Rule List Panel

- Data

  Invokes ISPF option 3.1 for the AOF test data set. This command is valid only if you entered a value into the Test Data Set Name field on the Specification Display panel.

- Globals

  Causes the Display Global Variables panel to appear. You can use the Display Global Variables panel to:

  Display a global variable's subnodes.

  Drop a node.

  Remove a node and its subnodes.

  Create and modify global variables.

- Locate rule

  Scrolls the panel so that the line referring to rule is the top line on the panel.

- OpsBrw

  Invokes the OPSLOG Browse Test Data panel. This panel is a full-screen display of current rule test data.

- Rules

  Invokes ISPF option 3.1 for the AOF test rule data set.

- Select rulename

  Selects rulename so that you can edit it.

- SORT columnname

  Sorts the specified column in default order. For example, you can issue this command to sort the rules according to the date and time that they were last modified:

  ```
  SORT CHANGED
  ```

  The default order varies by column.

- SORTA columnname

  Sorts the specified column in ascending order.

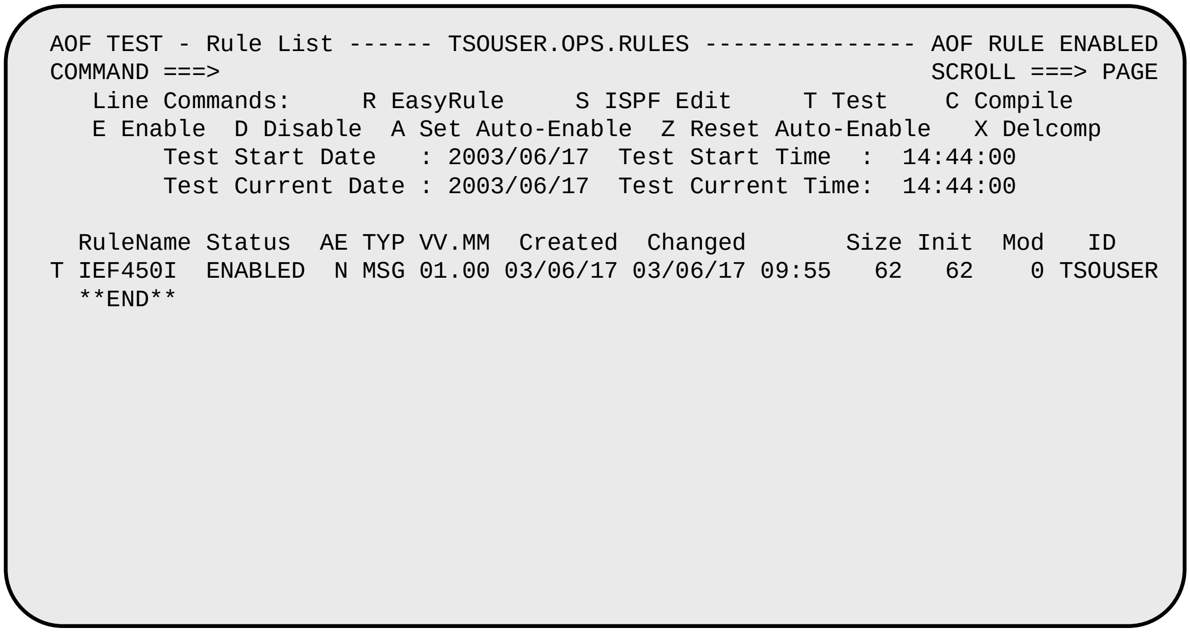
- SORTD columnname

  Sorts the specified column in descending order.

Line Commands for the Rule List Panel

- A - Sets the rule's auto-enable flag to Y. If you use the Rule Set List panel to enable the rule set to which this rule belongs, this rule is enabled. Contrast this command with the Z line command.
- B - Compiles the rule into the compiled rule data set.
- D - Disables a rule that was previously enabled.
- E - Enables a rule so that you can test it. If a rule contains syntax errors, the E command fails.
- R - Invokes EasyRule processing for the rule. You can use the R command for a rule only if you used EasyRule to create the rule and you have not used the S line command to edit the rule.
- S - Selects the rule for ISPF editing. The panel you see when you issue the S command is similar to an ISPF edit session of the rule. You may notice slight modifications to the panel which appear to remind you that you are within the AOF edit option. You can use the ISPF HELP command when you are within the editing session.
- T - Displays the AOF Test panel.
- X - Deletes the rule from the compiled rule data set.
- Z - Sets the rule's auto-enable flag to N. Even if you use the Rule Set List panel to enable the rule set to which this rule belongs, this rule will not be enabled. Contrast this command with the A line command.

Select for Testing



```
AOF TEST - Rule List ------ TSOUSER.OPS.RULES --------------- AOF RULE ENABLED
COMMAND ===>                                                  SCROLL ===> PAGE
   Line Commands:     R EasyRule     S ISPF Edit     T Test    C Compile
   E Enable  D Disable  A Set Auto-Enable  Z Reset Auto-Enable   X Delcomp
        Test Start Date   : 2003/06/17  Test Start Time  :  14:44:00
        Test Current Date : 2003/06/17  Test Current Time:  14:44:00

  RuleName Status  AE TYP VV.MM  Created  Changed       Size Init  Mod   ID
T IEF450I  ENABLED  N MSG 01.00 03/06/17 03/06/17 09:55   62   62    0 TSOUSER
  **END**
```

6 - 10

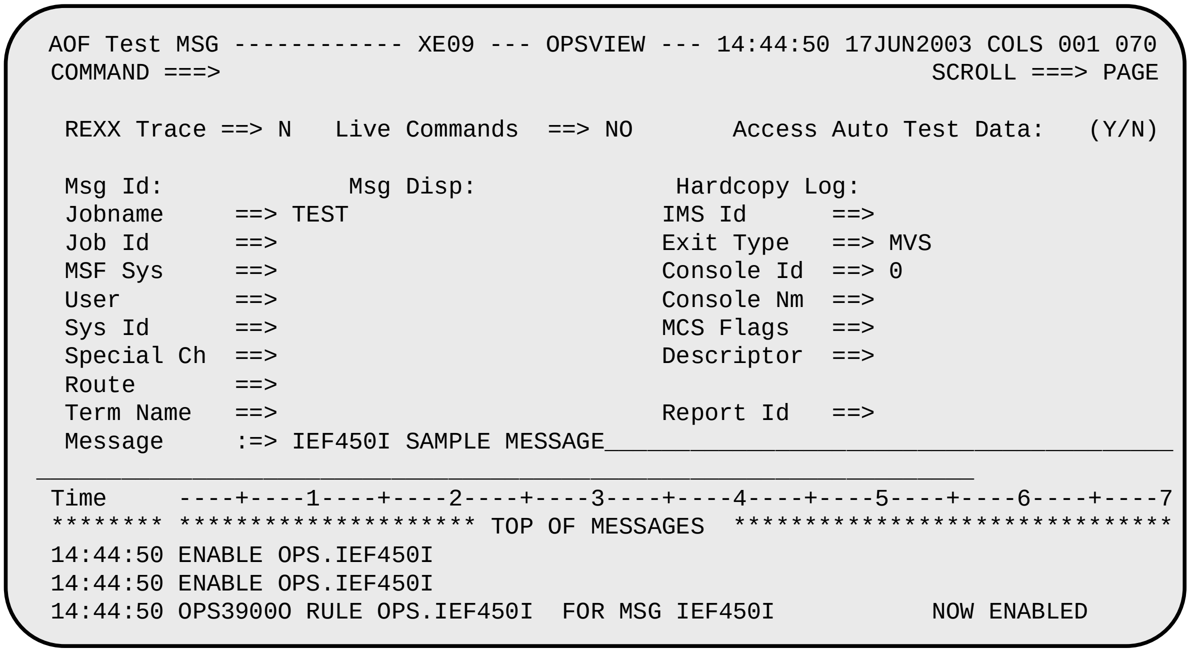
Select for Testing

After you enable a rule, you will notice that two fields on the Rule List panel (as shown in the example screen above) change. 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.

Once a rule is enabled, you can select it for testing.

Action: To select a rule for testing, type a T (Test) to the left of the desired rule name and then press Enter.

Specify Testing Criteria



```
AOF Test MSG ------------ XE09 --- OPSVIEW --- 14:44:50 17JUN2003 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     ==> TEST                      IMS Id      ==>
 Job Id      ==>                           Exit Type   ==> MVS
 MSF Sys     ==>                           Console Id  ==> 0
 User        ==>                           Console Nm  ==>
 Sys Id      ==>                           MCS Flags   ==>
 Special Ch  ==>                           Descriptor  ==>
 Route       ==>
 Term Name   ==>                           Report Id   ==>
 Message     :=> IEF450I SAMPLE MESSAGE________________________________________
_____________________________________________________________________
Time     ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
******** ******************** TOP OF MESSAGES  ******************************
14:44:50 ENABLE OPS.IEF450I
14:44:50 ENABLE OPS.IEF450I
14:44:50 OPS3900O RULE OPS.IEF450I  FOR MSG IEF450I           NOW ENABLED
```

Specify Testing Criteria

When you select a rule for testing, the AOF takes you to a test panel for the type of rule you selected. AOF Test panels prompt you for information about the rule you want to test. The example panel shown above, the AOF Test MSG panel, displays when a message rule is selected for testing.

The messages that appear in the message section at the bottom of the panel shown above came from OPSLOG. They indicate that the rule named IEF450I stored within the OPS rule set is enabled.

Action: Typically, you would enter values on an AOF Test MSG panel as follows:

1. Type the message ID and any text in the Message field. A message rule test will not run unless this field has been completed. For example, on the example panel above, IEF450I is entered in the Message field.
2. Type characters in the other fields as required to satisfy the conditions specified in the rule. For example, on the example panel above, TEST is entered in the Jobname field.
3. Press Enter to run a test message against the enabled rule.

When you finish typing data into an AOF Test panel and press Enter, the AOF tries to match the information you specified on the panel to the AOF's enabled rules. If rules other than the one selected for testing have been enabled, they will be included in the test. The AOF displays the test results in the bottom part of the AOF Test panel. Once you see the results, you can press PF3 to terminate the rule test session. If you want to continue testing, you can change some of the field data and press Enter again.

Common AOF Test Panel Fields

Although there is a different AOF Test panel for each type of rule, the panels are very similar. In fact, many of the same fields appear on all AOF Test panels. Below are the common AOF Test panel fields; see your OPS/MVS documentation for information about the fields on specific AOF Test panels.

- Access Auto Test Data

  If you have already extracted data from OPSLOG Browse to use for your rule test and you enter a Y in this field, the AOF displays the AOF Test Data Selection panel, where you can view the extracted data.

  If you have not extracted data and you enter a Y, the AOF displays the OPSLOG Browse Test Data panel so that you can extract data.

- Console Id/Console Number

  An arbitrary number from 1 to 32767 that is associated with the simulated console being used to enter the command or send the message.

- Console Name/Console Nm

  The name of the simulated console being used to enter the command or send the message.

- Exit Type

  The simulated exit type associated with the current command. Types are IMS, JES3, OS/390, and OMG.

- IMS ID

  The four-character ID of the IMS system being simulated.

- Jobname

  For a message rule test, the simulated job name (TSO user ID or task name) associated with the current message.

  For a command rule test or a request rule test, the simulated job name (TSO user ID or task name) associated with the current command.

For an end-of-memory rule test, the name of the test. The name of the test can be any valid job name or the wildcard character.

- Live Commands

  A value indicating how the AOF should treat host commands during the rule test. If the value is YES, commands are issued on your system. If the value is NO, commands are not issued, but they are simulated for test purposes.

- Report Id

  For events originating in the generic data set interface or the OMEGAMON interface, this value indicates the report ID associated with the event.

- REXX Trace

  A value indicating if and how the AOF should trace a command that REXX executes. Common values are:

  N – Normal. The AOF traces only those host commands that fail.

  R – Results. The AOF traces all clauses before execution, along with the final results. This is useful for general debugging.

  I – Intermediates. Similar to R, but the AOF also traces all REXX clauses and intermediate results.

  For details about tracing possibilities, refer to M.F. Cowlishaw's The REXX Language: A Practical Approach to Programming. You can order a copy of Cowlishaw's book from Prentice-Hall.

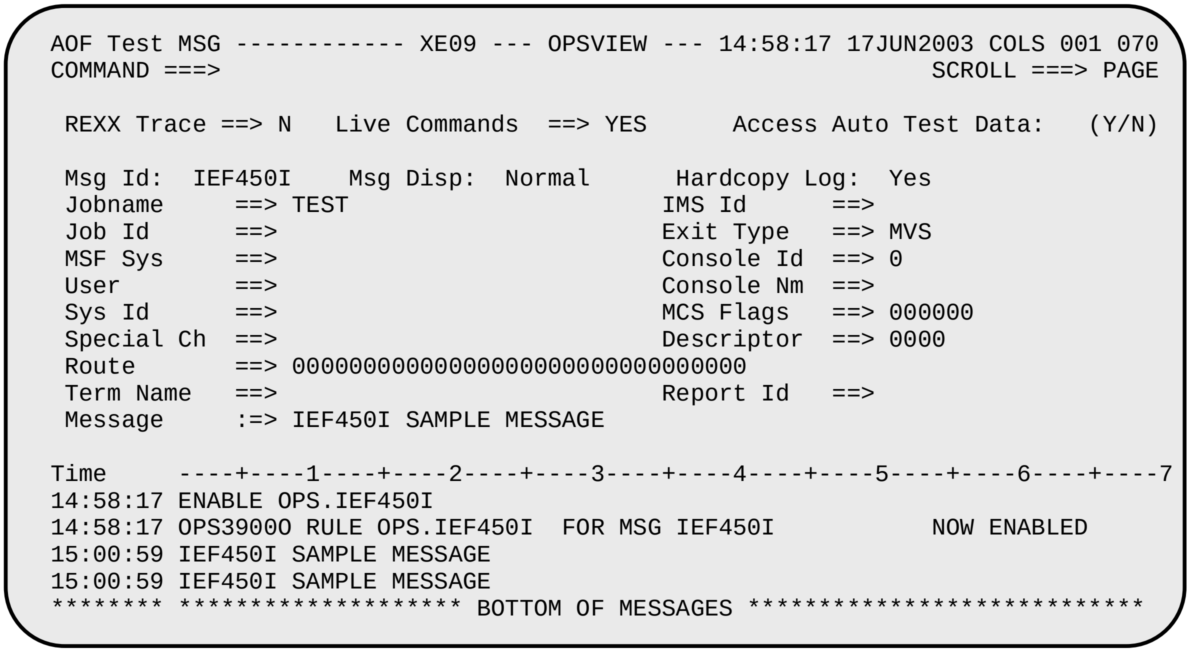
- Sys ID/System ID

  The identification string for the system.

- User/User Field

  Specifies eight bytes of data you want to pass between rules that fire for the same event. Depending on what type of rule you are testing, the event can be a command, message, global variable event, OMEGAMON event message, request event, end-of-memory event, or delete-operator-message event.

Test with Live Commands



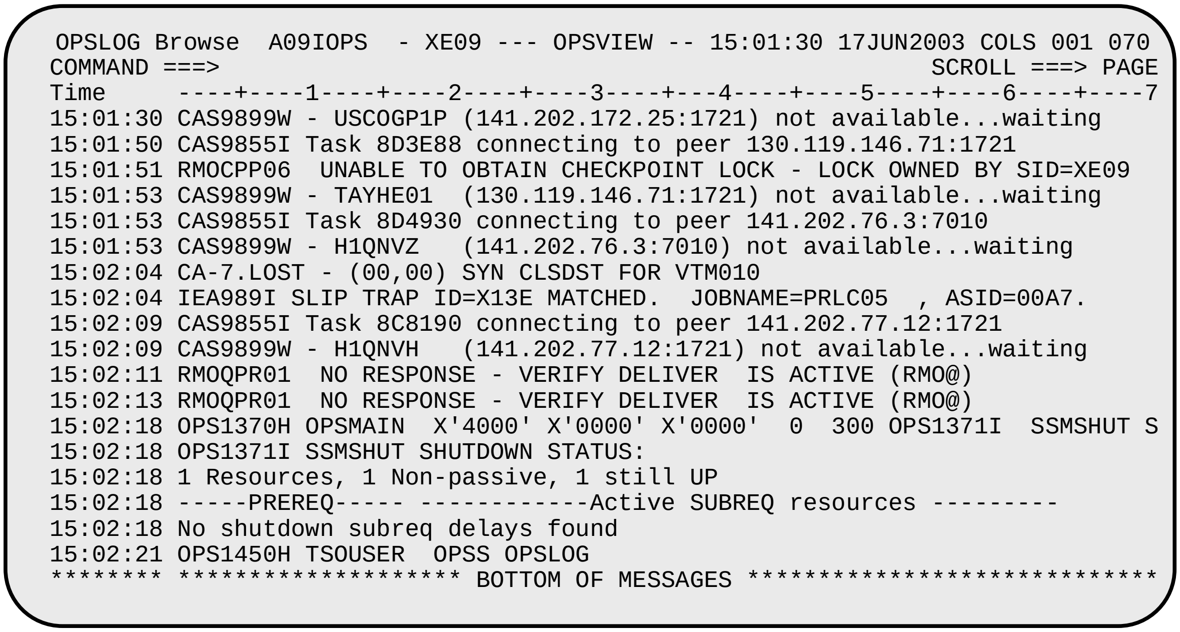
```
AOF Test MSG ------------ XE09 --- OPSVIEW --- 14:58:17 17JUN2003 COLS 001 070
COMMAND ===>                                                   SCROLL ===> PAGE

 REXX Trace ==> N   Live Commands  ==> YES       Access Auto Test Data:   (Y/N)

 Msg Id:  IEF450I    Msg Disp:  Normal       Hardcopy Log:  Yes
 Jobname     ==> TEST                       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

Time     ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
14:58:17 ENABLE OPS.IEF450I
14:58:17 OPS3900O RULE OPS.IEF450I  FOR MSG IEF450I            NOW ENABLED
15:00:59 IEF450I SAMPLE MESSAGE
15:00:59 IEF450I SAMPLE MESSAGE
******** ******************** BOTTOM OF MESSAGES ****************************
```

Test with Live Commands

To test with live commands, or those that are issued on your system, specify YES in the Live Commands field of an AOF Test panel as shown above.

The Live Commands field tells the AOF how it should treat host commands during the rule test. If the value is YES, commands are issued on your system. If the value is NO, commands are not issued; rather, they are simulated for test purposes.

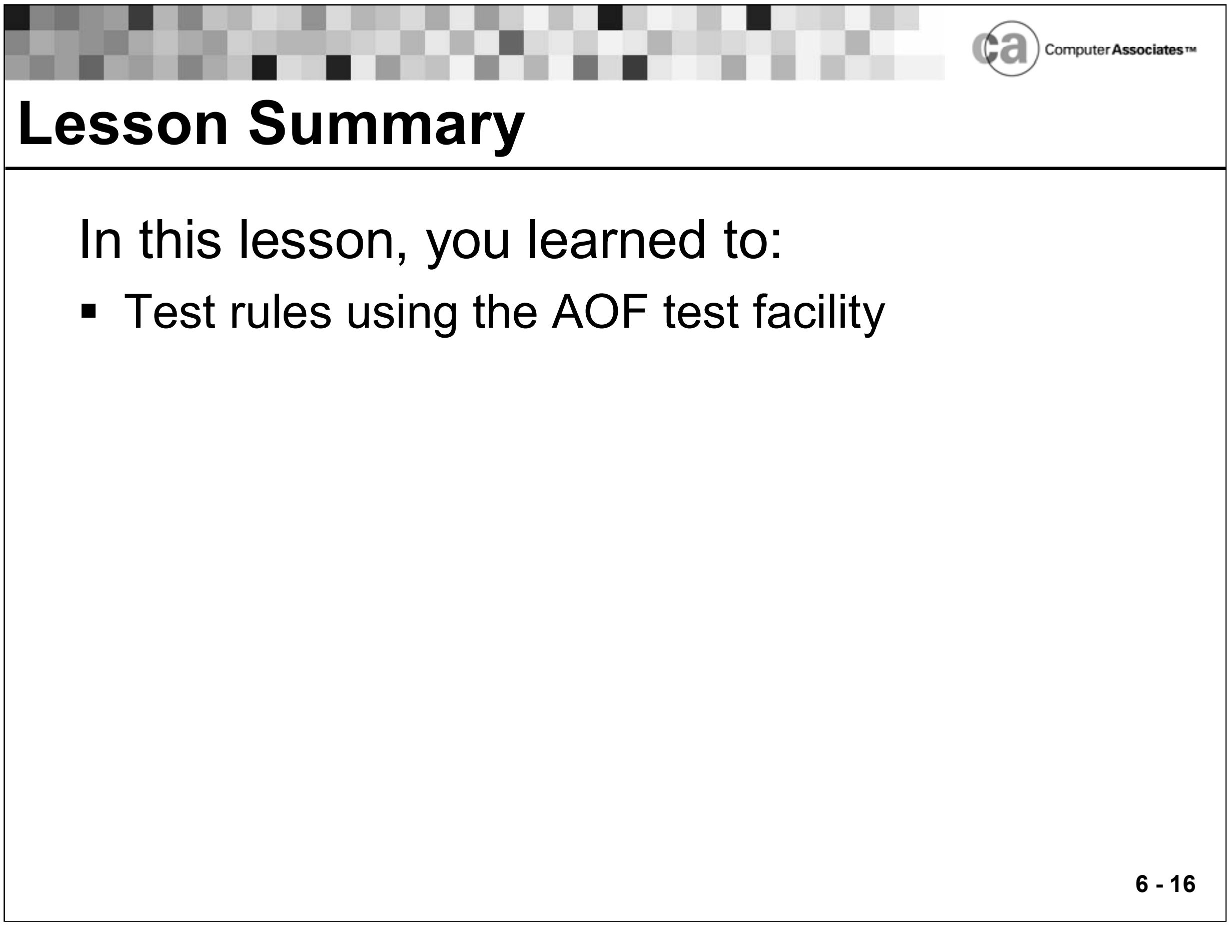
View Results



```
 OPSLOG Browse  A09IOPS  - XE09 --- OPSVIEW -- 15:01:30 17JUN2003 COLS 001 070
COMMAND ===>                                                  SCROLL ===> PAGE
Time     ----+----1----+----2----+----3----+---4----+----5----+----6----+----7
15:01:30 CAS9899W - USCOGP1P (141.202.172.25:1721) not available...waiting
15:01:50 CAS9855I Task 8D3E88 connecting to peer 130.119.146.71:1721
15:01:51 RMOCPP06  UNABLE TO OBTAIN CHECKPOINT LOCK - LOCK OWNED BY SID=XE09
15:01:53 CAS9899W - TAYHE01  (130.119.146.71:1721) not available...waiting
15:01:53 CAS9855I Task 8D4930 connecting to peer 141.202.76.3:7010
15:01:53 CAS9899W - H1QNVZ   (141.202.76.3:7010) not available...waiting
15:02:04 CA-7.LOST - (00,00) SYN CLSDST FOR VTM010
15:02:04 IEA989I SLIP TRAP ID=X13E MATCHED.  JOBNAME=PRLC05  , ASID=00A7.
15:02:09 CAS9855I Task 8C8190 connecting to peer 141.202.77.12:1721
15:02:09 CAS9899W - H1QNVH   (141.202.77.12:1721) not available...waiting
15:02:11 RMOQPR01  NO RESPONSE - VERIFY DELIVER  IS ACTIVE (RMO@)
15:02:13 RMOQPR01  NO RESPONSE - VERIFY DELIVER  IS ACTIVE (RMO@)
15:02:18 OPS1370H OPSMAIN  X'4000' X'0000' X'0000'  0  300 OPS1371I  SSMSHUT S
15:02:18 OPS1371I SSMSHUT SHUTDOWN STATUS:
15:02:18 1 Resources, 1 Non-passive, 1 still UP
15:02:18 -----PREREQ----- ------------Active SUBREQ resources ---------
15:02:18 No shutdown subreq delays found
15:02:21 OPS1450H TSOUSER  OPSS OPSLOG
******** ******************* BOTTOM OF MESSAGES ****************************
```

View Results

To view the results of your live command test, access OPSLOG Browse.



Lesson Summary

In this lesson, you learned to:

- Test rules using the AOF test facility

6 - 16

Lesson 6 Activity

6 - 17

Lesson 6 Activity - Testing Rules

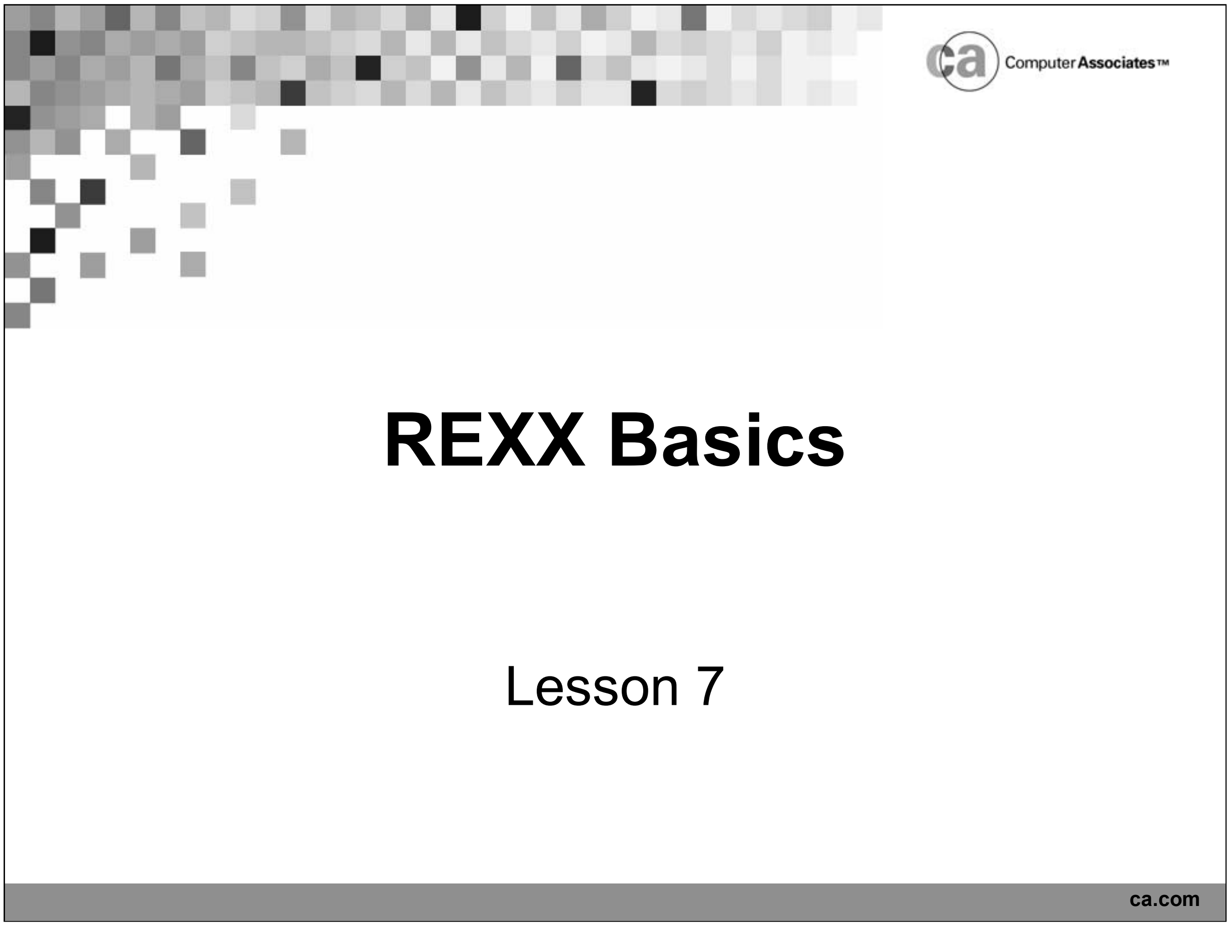
Task

- Use the OPSVIEW Editors option to test the rules named NOTCTLG and MNSTATUS, which you created in Activity 2.2.

Steps to Take

1. Access the AOF Test Rule List panel.
2. Enable the rule and select it for testing.
3. Test the rule.
4. Check the test results.

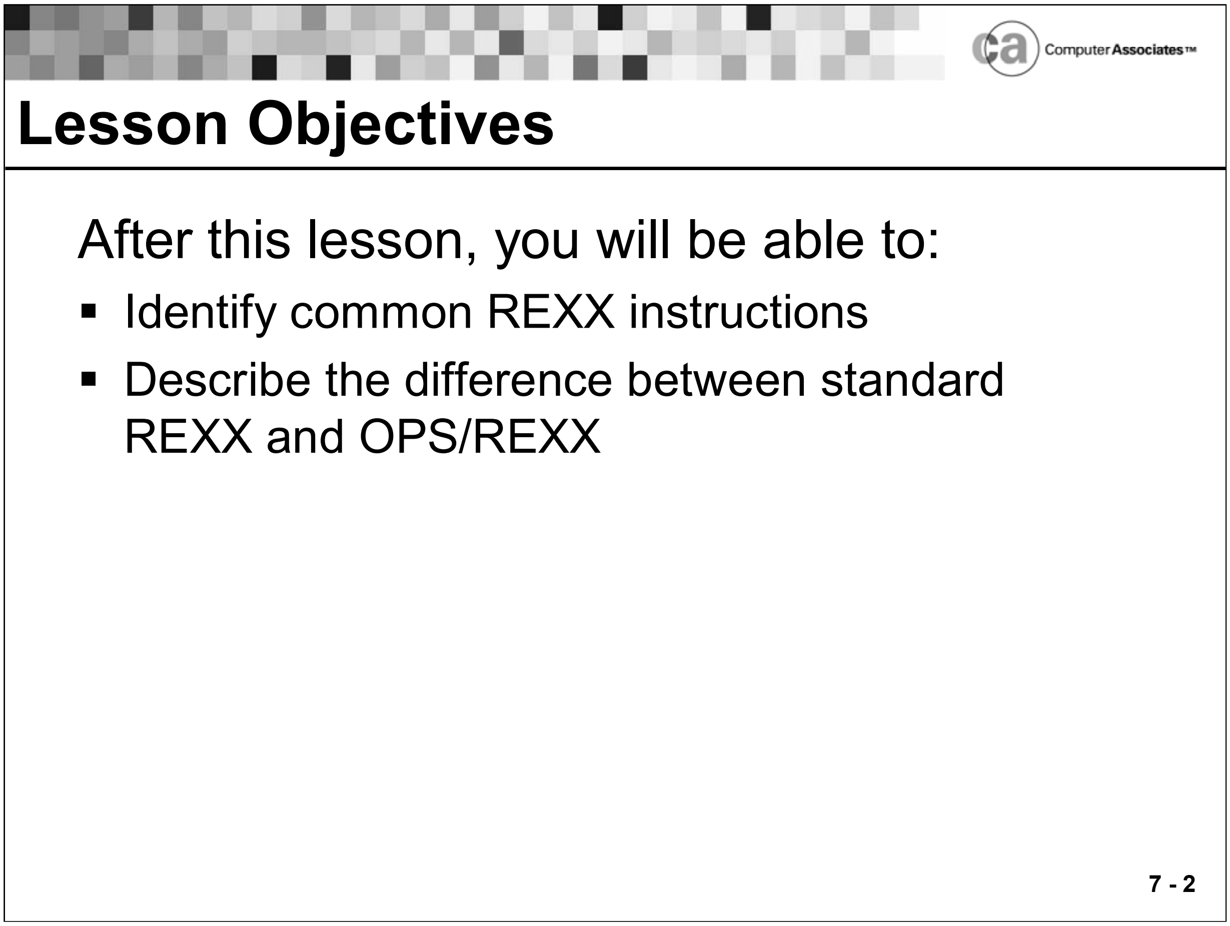
Notes:



REXX Basics

Lesson 7

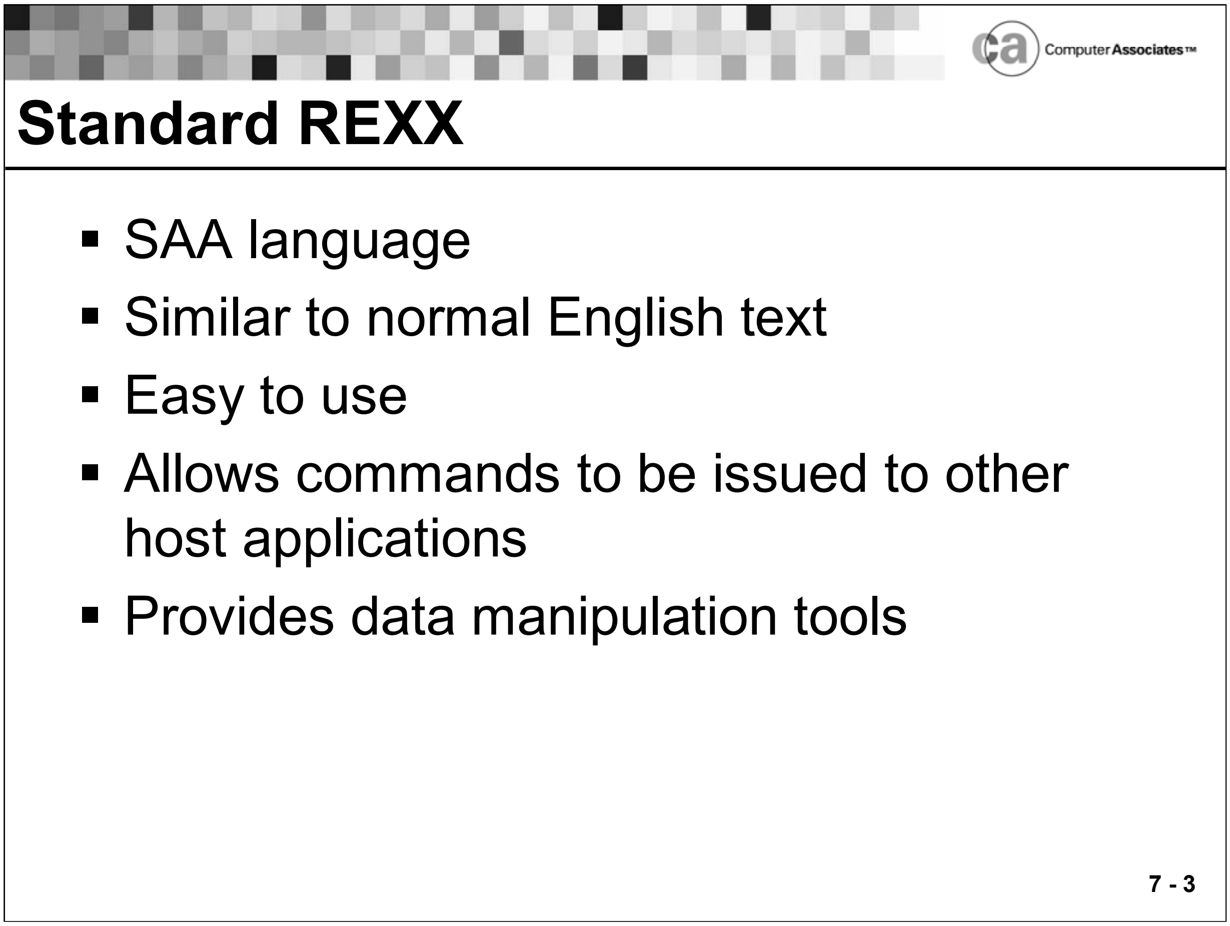
ca.com



Lesson Objectives

After this lesson, you will be able to:

- Identify common REXX instructions
- Describe the difference between standard REXX and OPS/REXX



Standard REXX

- SAA language
- Similar to normal English text
- Easy to use
- Allows commands to be issued to other host applications
- Provides data manipulation tools

7 - 3

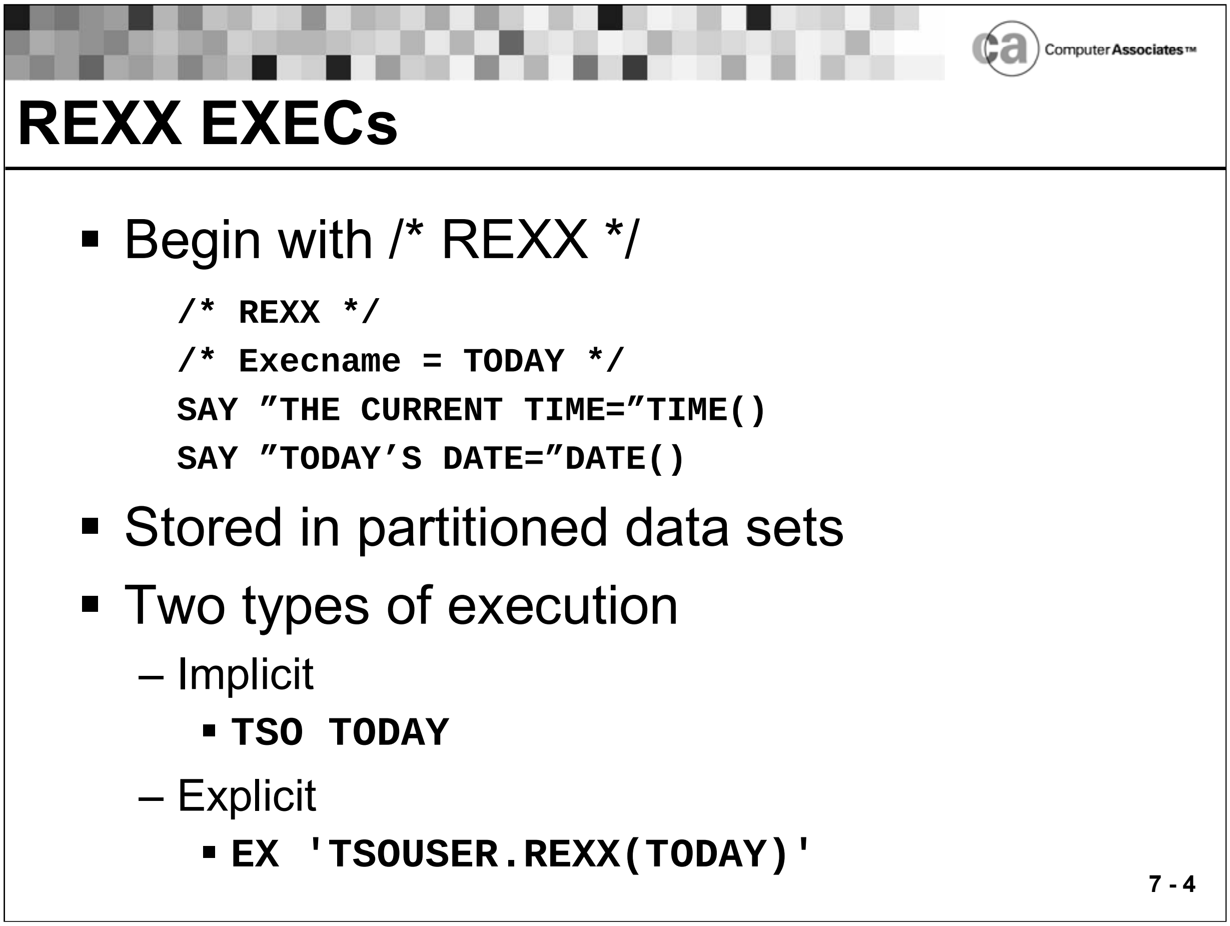
Standard REXX

REXX, which stands for Restructured EXtended eXecutor, is the standard common language for all of IBM’s environments under its Systems Application Architecture (SAA). The REXX language was developed by Michael Cowlishaw, whose book, THE REXX LANGUAGE: A Practical Approach to Programming, is referred to throughout this lesson. REXX is the most powerful and easiest to use command language available.

Since REXX is similar to normal English text, both casual users and computer professionals can use it with ease. REXX enables you to manipulate familiar objects such as words, numbers, and names.

REXX provides a robust set of operators and functions that allow you to manipulate your data naturally. It even allows you to issue commands to host environments.

Within your AOF rules, you can use standard REXX programming tools to make decisions about an AOF event or further break down the data about an event.



REXX EXECs

- Begin with /* REXX */

```
/* REXX */
/* Execname = TODAY */
SAY ”THE CURRENT TIME=”TIME()
SAY ”TODAY’S DATE=”DATE()
```

- Stored in partitioned data sets
- Two types of execution
  - Implicit
    - **TSO TODAY**
  - Explicit
    - **EX 'TSOUSER.REXX(TODAY)'**

7 - 4

REXX EXECs

All REXX EXECs are composed of a sequence of clauses. Clauses can span multiple lines, and multiple clauses can exist on one line. When writing REXX EXECs, you should begin them with the comment /* REXX */.

REXX EXECs are stored in partitioned data sets that are allocated via the SYSEXEC DD concatenation. For example:

```
//SYSEXEC DD DISP=SHR,DSN=OPSMVS.EXECS
          DD DISP=SHR,DSN=TSOUSER.REXX
```

REXX EXECs can be executed in the following ways:

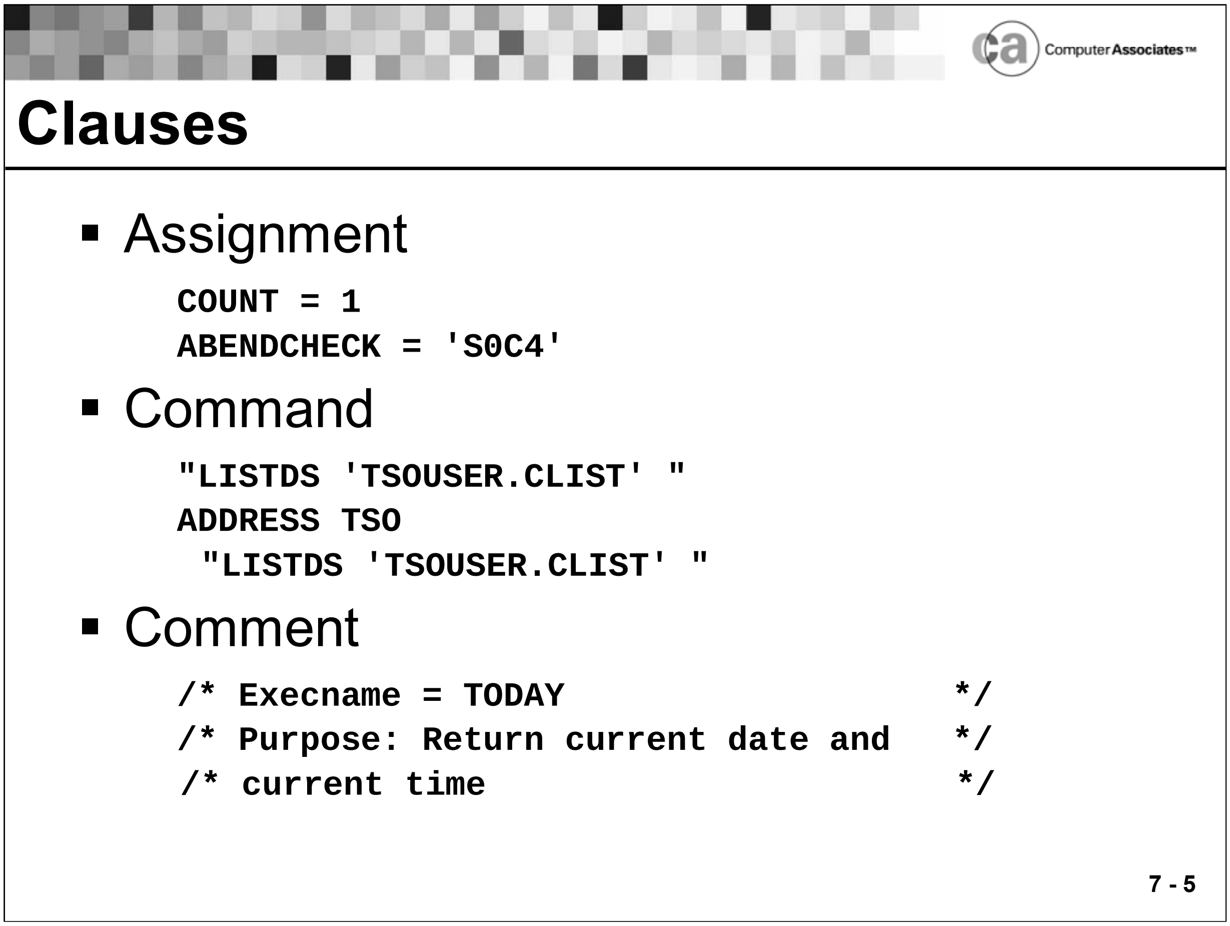
- Implicitly, by specifying the member name on the command line in TSO or by specifying the member name via TSO option 6. For example:

  ```
  TSO TODAY
  ```

  TSO searches the SYSEXEC concatenation for the member name you specified.

- Explicitly, by specifying the name of the data set that contains the member. For example:

  ```
  EX 'TSOUSER.REXX(TODAY)'
  ```



Clauses

- Assignment

```
COUNT = 1
ABENDCHECK = 'S0C4'
```

- Command

```
"LISTDS 'TSOUSER.CLIST' "
ADDRESS TSO
 "LISTDS 'TSOUSER.CLIST' "
```

- Comment

```
/* Execname = TODAY                     */
/* Purpose: Return current date and     */
/* current time                         */
```

7 - 5

Clauses

REXX programs are built upon clauses. A clause is the grouping of REXX syntax.

Clauses are composed of the following:

- Zero or more blanks (ignored).
- A series of tokens.
- Zero or more blanks (ignored).
- A semicolon (;) delimiter that may be indicated by the end of the line, selected keywords, or the colon (:).

Types of Clauses

- Assignment – A clause in the form symbol = expression. An assignment provides a variable with its value.

  Examples:

  This example illustrates assigning a variable called COUNT the value of 1:

  ```
  COUNT = 1
  ```

  This example shows assigning a variable called ABENDCHECK the value 'S0C4':

  ```
  ABENDCHECK = 'S0C4'
  ```

- Command – A clause that consists of an expression only.

  Examples:

  These examples demonstrate two ways in which you can issue a LISTDS TSO command:

  To send to the default address environment

  ```
  "LISTDS 'TSOUSER.CLIST'"
  ```

  To first set the intended address environment

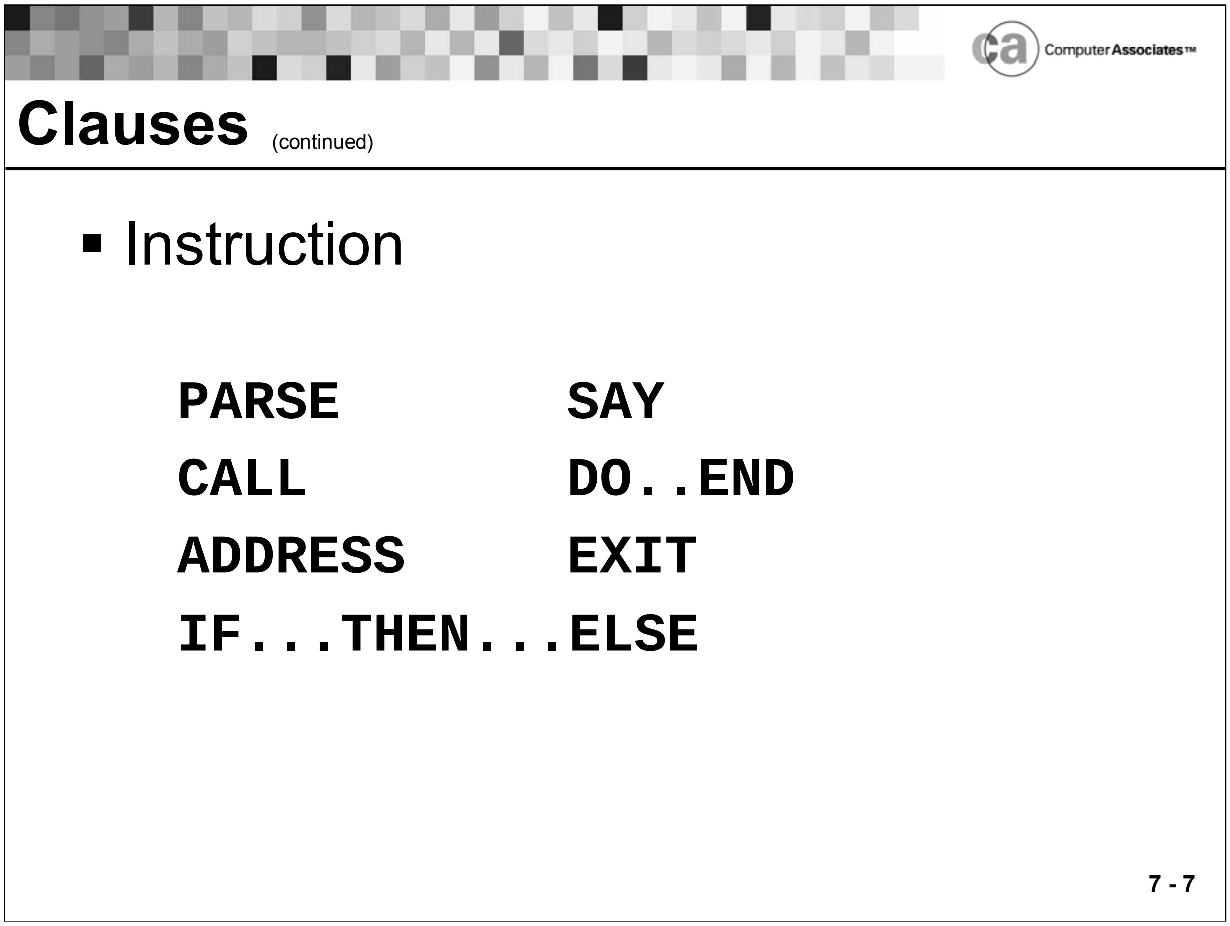
  ```
  ADDRESS TSO
      "LISTDS 'TSOUSER.CLIST'"
  ```

- Comment – A statement used to document a program. Comments have no impact on the execution of a program; rather, they are used to explain certain aspects of a program (for example, information that may be helpful in running a job).

  Comments begin with “/*” and end with “*/”. You may use any characters within these delimiters.

  Examples:

  ```
  /* Execname = TODAY */
  /* Purpose: Return current date and current
  time */
  ```



Clauses (continued)

- Instruction

```
PARSE          SAY
CALL           DO..END
ADDRESS        EXIT
IF...THEN...ELSE
```

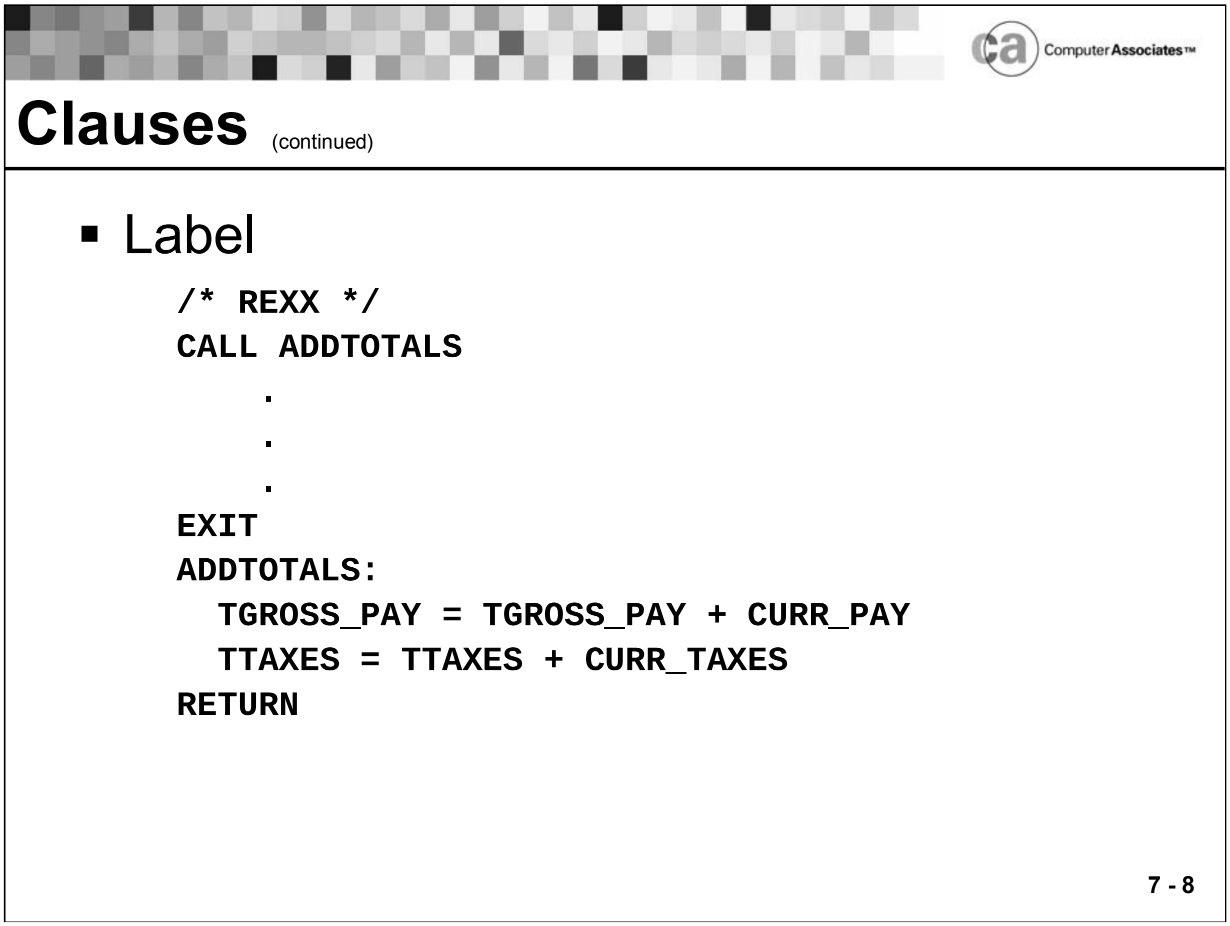
7 - 7

Types of Clauses

- Instruction – One or more clauses that describe an action to be taken. (Instructions are described in detail later.)

  Examples:

```
PARSE                SAY
CALL                 DO…END
ADDRESS              EXIT
IF...THEN...ELSE
```



Clauses (continued)

- Label

```
/* REXX */
CALL ADDTOTALS
    .
    .
    .
EXIT
ADDTOTALS:
  TGROSS_PAY = TGROSS_PAY + CURR_PAY
  TTAXES = TTAXES + CURR_TAXES
RETURN
```

7 - 8

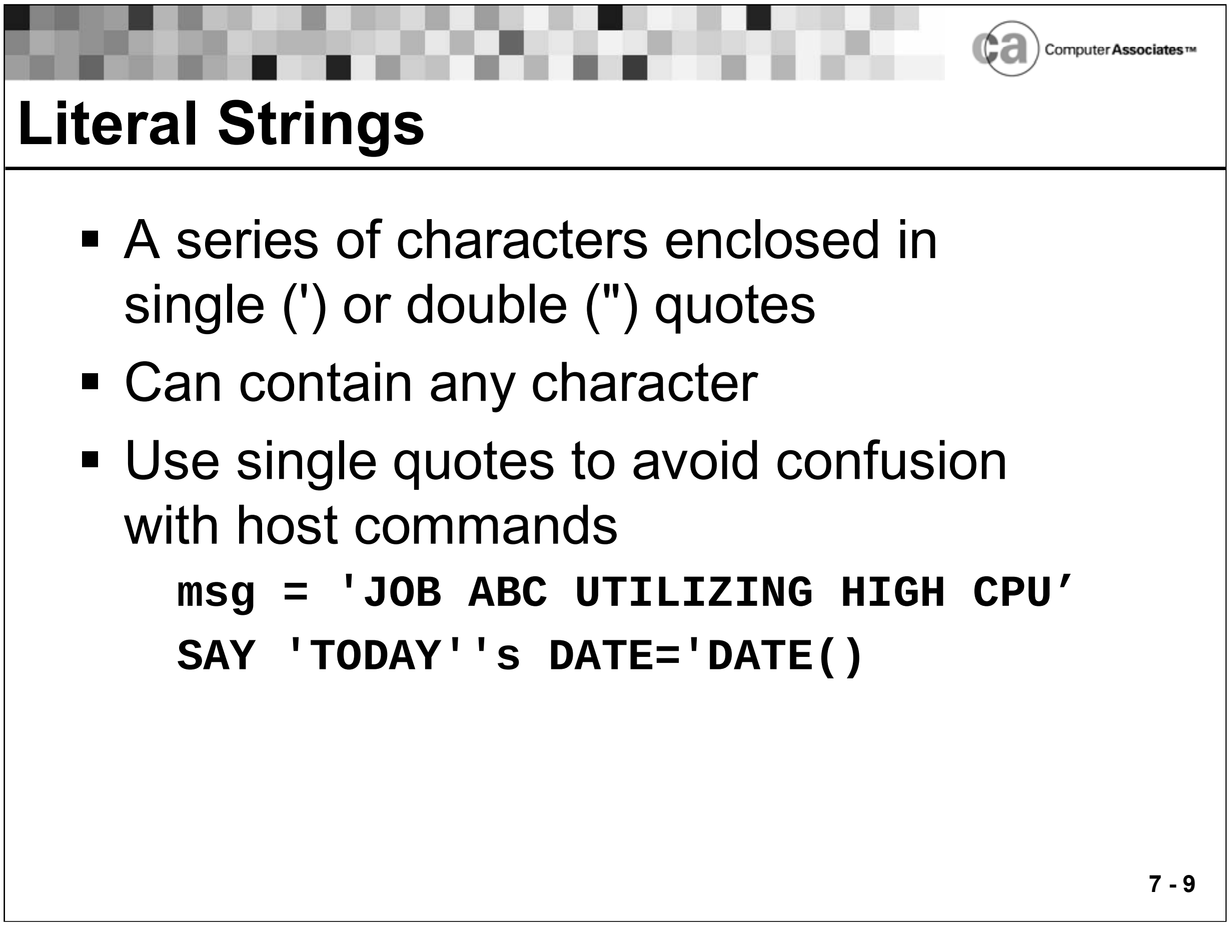
Types of Clauses (cont.)

- Label – A clause that consists of one symbol followed by a colon. In labels, the colon acts as a clause separator; therefore, no semicolon is required. Labels are primarily used to identify internal subroutines.

  Example

  The following is an example of REXX code that calls a subroutine named ADDTOTALS:

```
/* REXX */
CALL ADDTOTALS
    .
    .
    .
EXIT
ADDTOTALS:
 TGROSS_PAY = TGROSS_PAY + CURR_PAY
 TTAXES = TTAXES + CURR_TAXES
 RETURN
```



Literal Strings

- A series of characters enclosed in single (') or double (") quotes
- Can contain any character
- Use single quotes to avoid confusion with host commands

```
    msg = 'JOB ABC UTILIZING HIGH CPU’
    SAY 'TODAY''s DATE='DATE()
```

7 - 9

Literal Strings

A literal string is a series of characters that is enclosed in single (') or double (") quotes. Literal strings can contain any characters and cannot be modified.

Tip: You should enclose literal strings with single quotes to avoid confusing them with host commands.

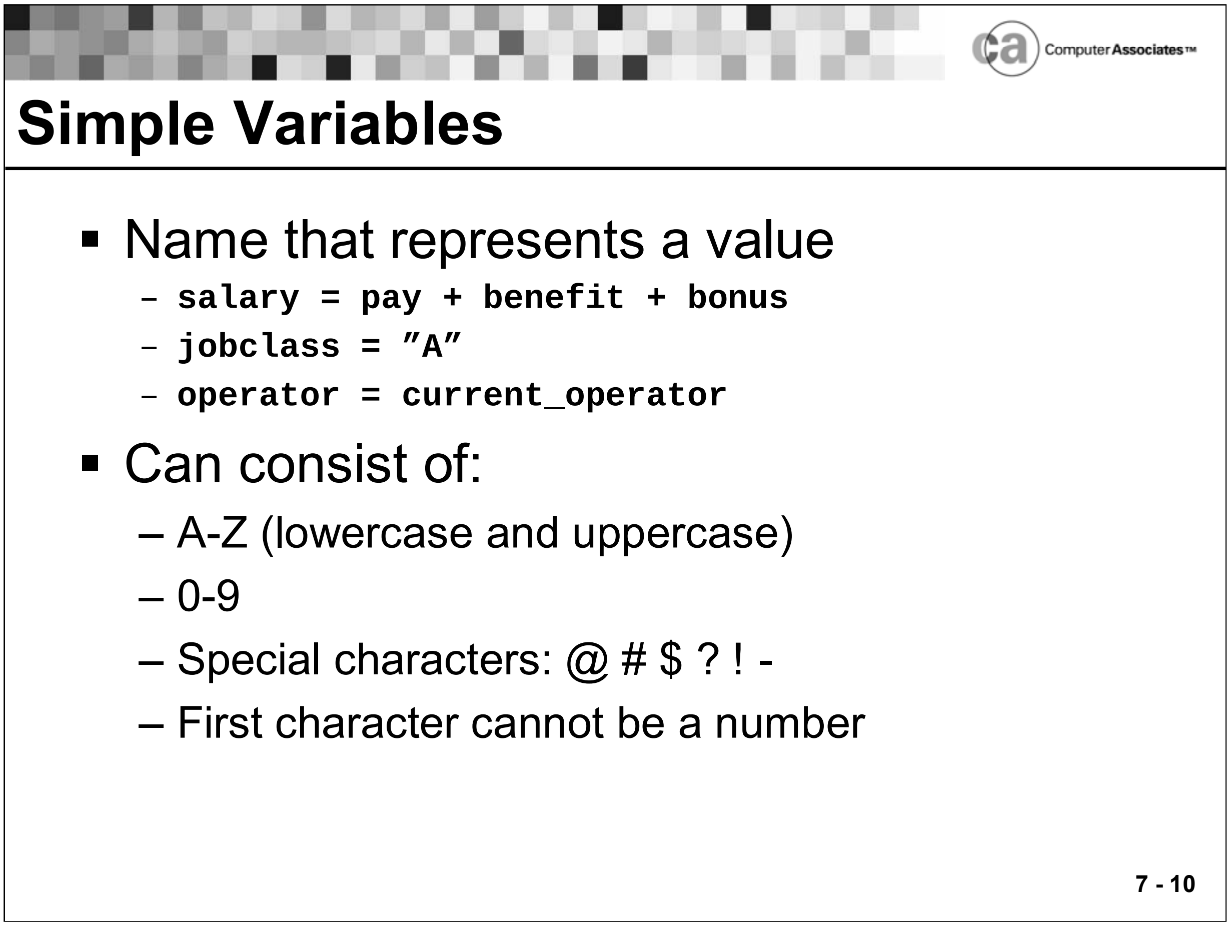
Examples

This example assigns a literal string to the variable “msg”:

```
msg = 'JOB ABC UTILIZING HIGH CPU'
```

This example illustrates using the REXX SAY instruction with the DATE function:

```
SAY 'TODAY''s DATE='DATE()
```



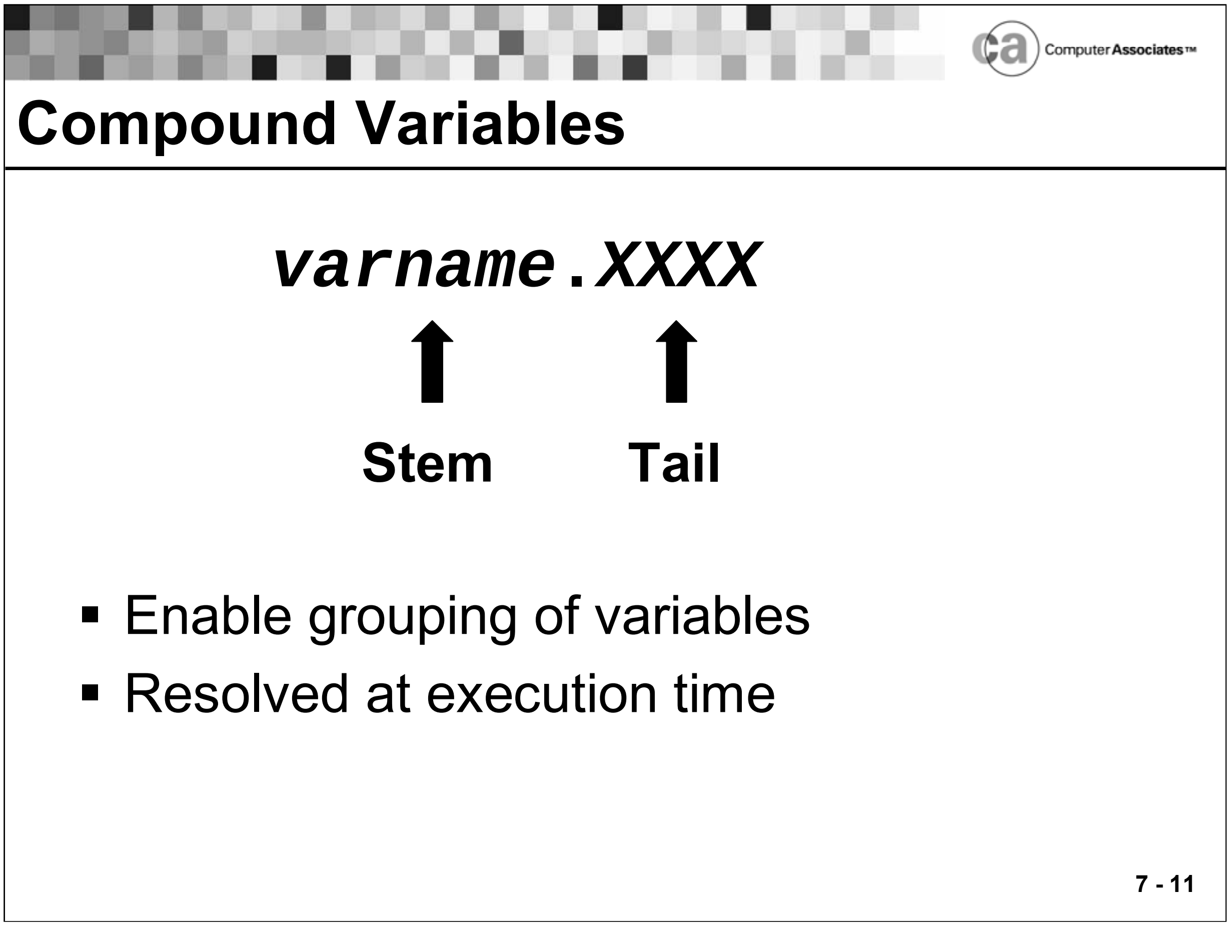
Simple Variables

- Name that represents a value
  - `salary = pay + benefit + bonus`
  - `jobclass = ”A”`
  - `operator = current_operator`
- Can consist of:
  - A-Z (lowercase and uppercase)
  - 0-9
  - Special characters: @ # $? ! -
  - First character cannot be a number

Simple Variables

The first character of a simple variable cannot be a number.

REXX converts variable names to uppercase. If you do not assign a value to a simple variable, REXX will assign a value equal to the variable name in uppercase.



Compound Variables

varname.XXXX

Stem Tail

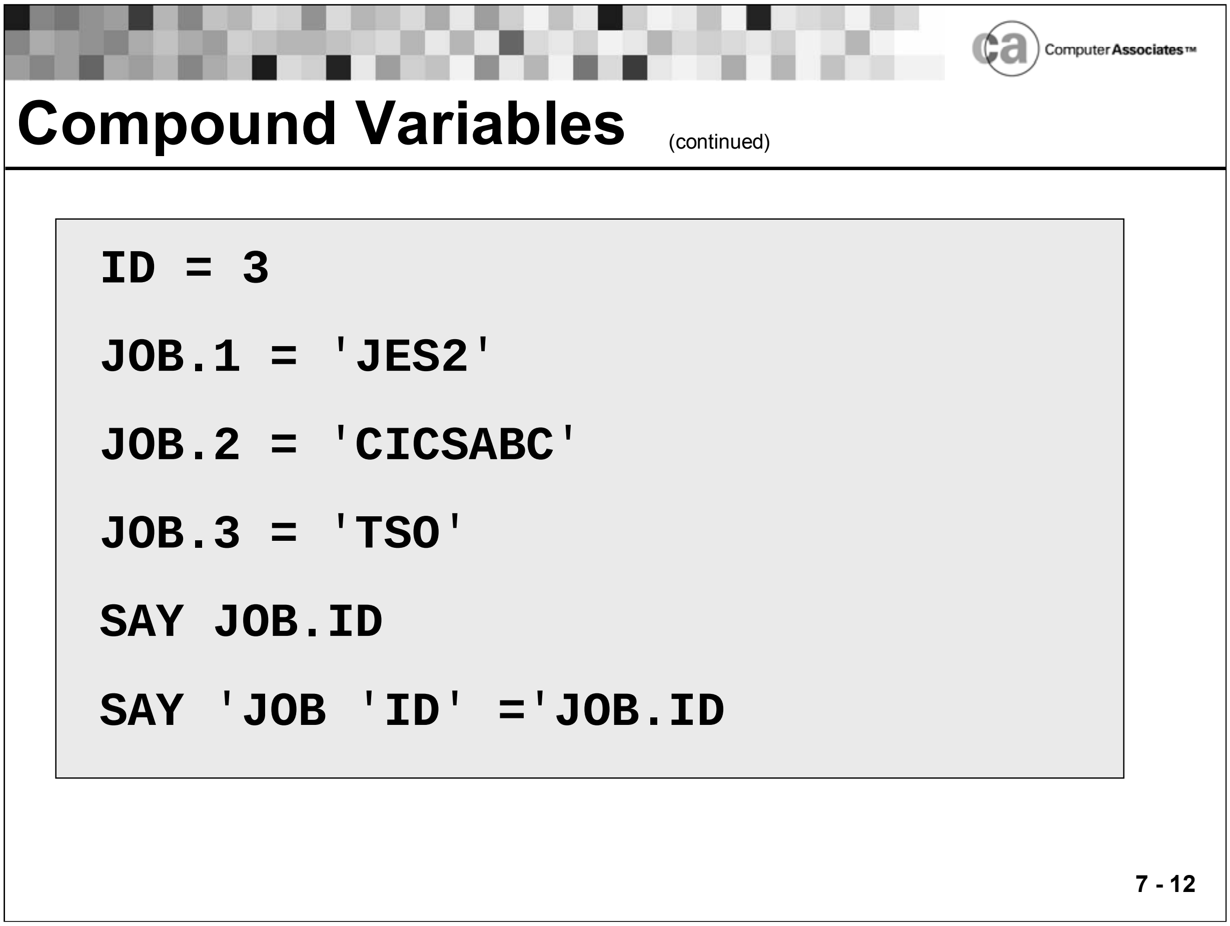
- Enable grouping of variables
- Resolved at execution time

Compound Variables

A compound variable enables the substitution of variables within its name, when it is referred to. Compound variables allow you to group variables, which is useful when creating tables and lists. They are resolved when the program is executed.

Compound variables contain at least one period and two other characters. They cannot start with a digit or a period. If there is only one period, it cannot be the last character.

The stem is the variable name, up to and including the first period. The stem is followed by the tail. (A tail can be composed of a constant symbol (0-9), a simple variable, or a null value.)



Compound Variables (continued)

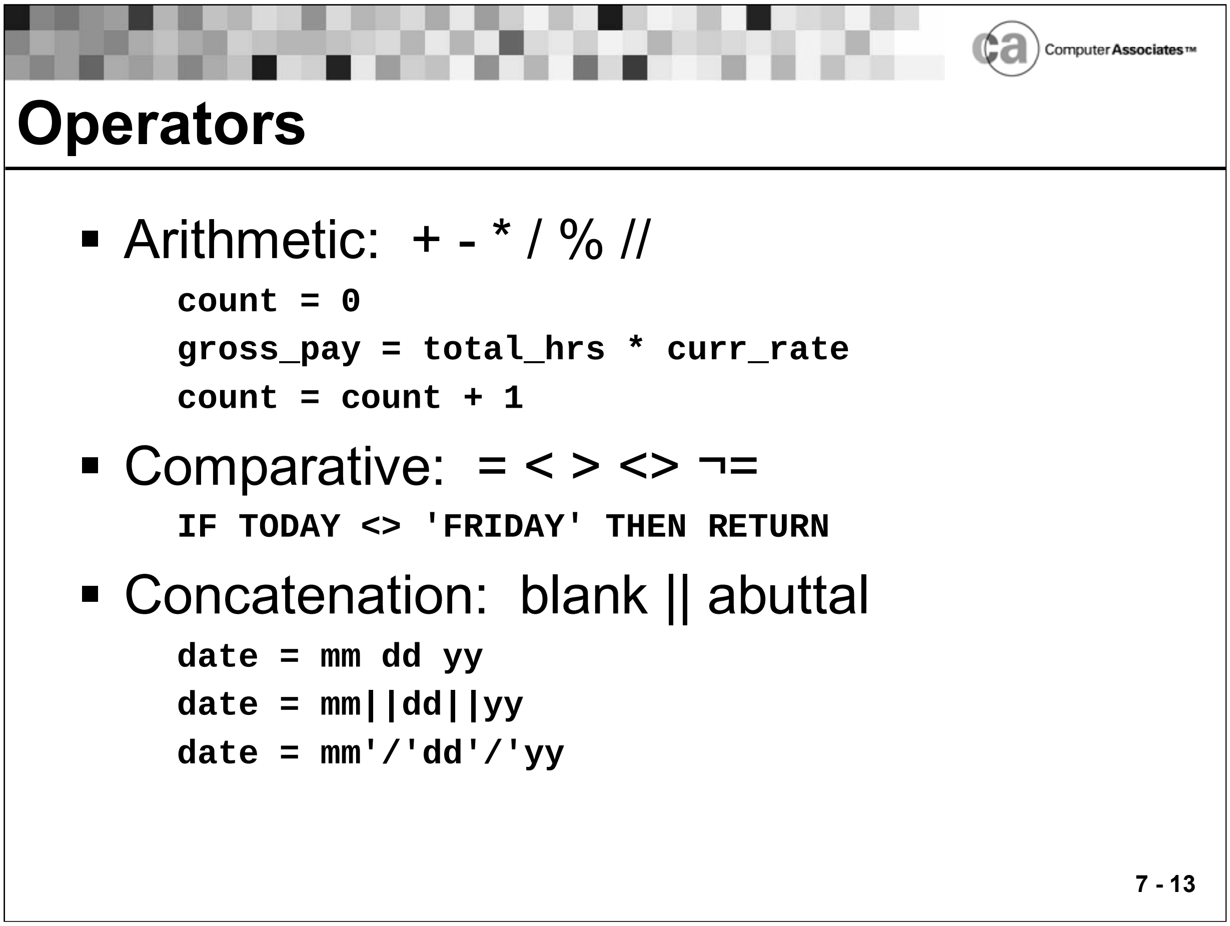
```
ID = 3

JOB.1 = 'JES2'

JOB.2 = 'CICSABC'

JOB.3 = 'TSO'

SAY JOB.ID

SAY 'JOB 'ID' ='JOB.ID
```

7 - 12

Compound Variables

Here are some examples of compound variables:

- `ID = 3`
- `JOB.1 = 'JES2'`
- `JOB.2 = 'CICSABC'`
- `JOB.3 = 'TSO'`
- `SAY JOB.ID`
- `SAY 'JOB 'ID' ='JOB.ID`



Operators

- Arithmetic: + - * / % //

```
count = 0
gross_pay = total_hrs * curr_rate
count = count + 1
```

- Comparative: = < > <> ¬=

```
IF TODAY <> 'FRIDAY' THEN RETURN
```

- Concatenation: blank || abuttal

```
date = mm dd yy
date = mm||dd||yy
date = mm'/'dd'/'yy
```

7 - 13

Operators

An operator is a symbol that indicates the action that is to be performed on operands.

Types of Operators

- Arithmetic – Character strings that are numbers can be combined using the following operators:

  + Add

  - Subtract

  * Multiply

  / Divide

  % Integer divide. This operator divides and returns the integer.

  // Remainder. This operator divides and returns the remainder.

Examples:

```
count = 0
gross_pay = total_hrs * curr_rate
count = count + 1
```

- Comparative – An operator that compares two terms and returns the value 1 if the result is true or 0 if it is not.

=	Equal
<	Less than
>	Greater than
<> or ¬=	Not equal

Example:

```
IF TODAY <> 'FRIDAY' THEN RETURN
```

- Concatenation – An operator that combines two strings to form one string by appending the second string to the end of the first string:

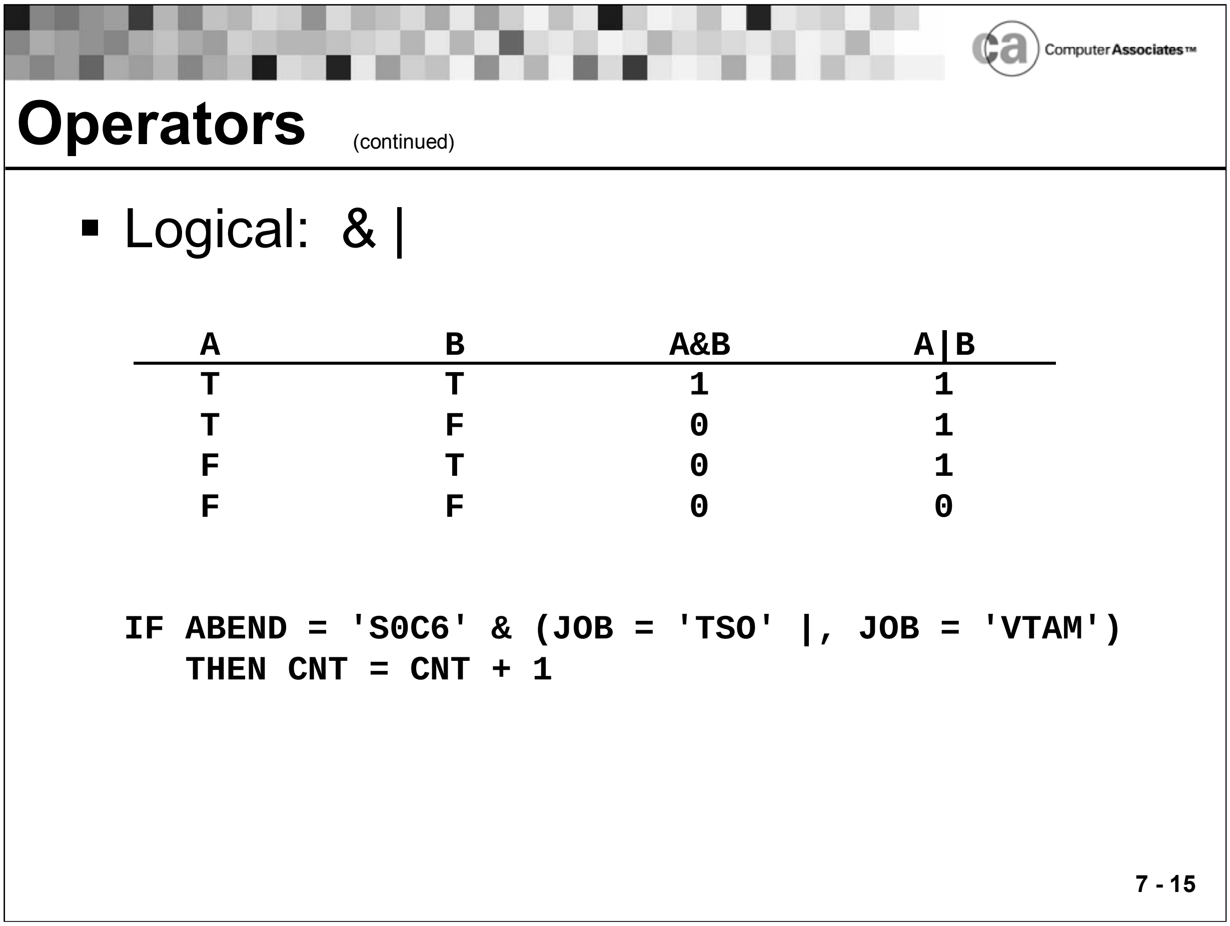
blank - Concatenate with one blank

| | or abuttal - Concatenate without an intervening blank

Note: An abuttal is where two terms in an expression are adjacent and are not separated by an operator.

Examples:

```
date = mm dd yy
date = mm||dd||yy
date = mm'/'dd'/'yy
```

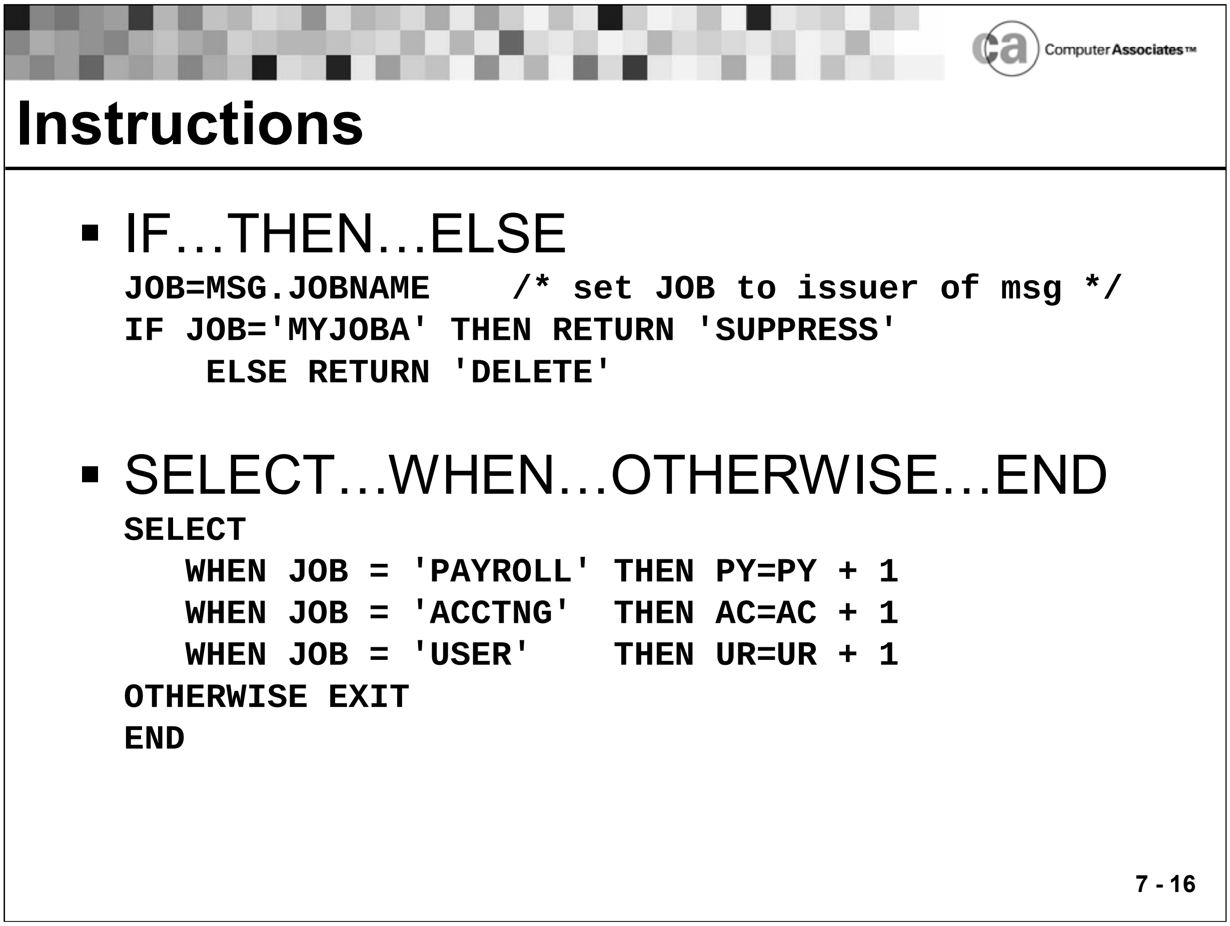


Operators (continued)

- Logical: & |

A	B	A&B	A|B
T	T	1	1
T	F	0	1
F	T	0	1
F	F	0	0

```
IF ABEND = 'S0C6' & (JOB = 'TSO' |, JOB = 'VTAM')
   THEN CNT = CNT + 1
```



Instructions

- IF…THEN…ELSE

```
JOB=MSG.JOBNAME    /* set JOB to issuer of msg */
IF JOB='MYJOBA' THEN RETURN 'SUPPRESS'
    ELSE RETURN 'DELETE'
```

- SELECT…WHEN…OTHERWISE…END

```
SELECT
   WHEN JOB = 'PAYROLL' THEN PY=PY + 1
   WHEN JOB = 'ACCTNG'  THEN AC=AC + 1
   WHEN JOB = 'USER'    THEN UR=UR + 1
OTHERWISE EXIT
END
```

7 - 16

Types of Instructions

- IF…THEN…ELSE

  Use this statement to conditionally execute an instruction or group of instructions or to select between two alternatives. It has the following form:

  IF expression [;] THEN [;] instruction [ELSE[;] instruction]

- SELECT…WHEN…OTHERWISE…END

  Use this statement to conditionally execute one of several alternative instructions. This statement is more efficient and effective than using a nested IF...THEN…ELSE statement.

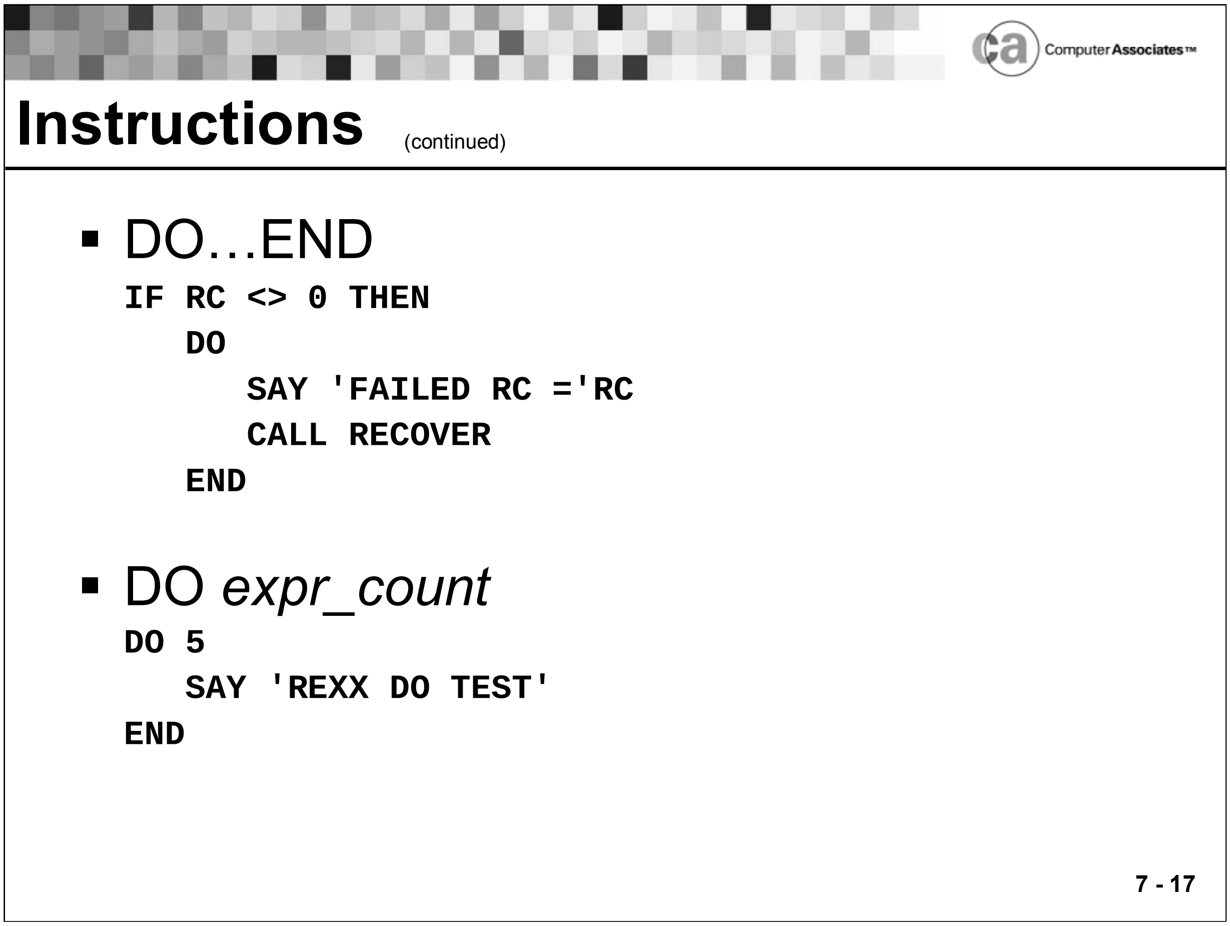
  It has the following form:

  ```
  SELECT ; whenlist [OTHERWISE [;]
       [instructionlist]] END ;
  ```

  Whenlist - One or more whenconstructs of the form:

  ```
       WHEN expression [;] THEN [;] instruction
  ```

  Instructionlist - A sequence of instructions.



Instructions (continued)

- DO…END

```
IF RC <> 0 THEN
   DO
      SAY 'FAILED RC ='RC
      CALL RECOVER
   END
```

- DO *expr_count*

```
DO 5
   SAY 'REXX DO TEST'
END
```

7 - 17

Types of Instructions

- DO…END

Use this statement to group instructions together. The instructions are executed once. It has the following form:

```
DO ;
  [instructionlist]
DO ;
```

Instructionlist - A sequence of instructions.

In the following example, when a return code is greater or less than 0 (zero) the rule logic executes a set of instructions once:

```
IF RC <> 0 THEN
  DO
    SAY 'FAILED RC ='RC
    CALL RECOVER
  END
```

- DO expr_count

  Use this statement to group instructions together and execute them repetitively (in a loop). It has the following form:

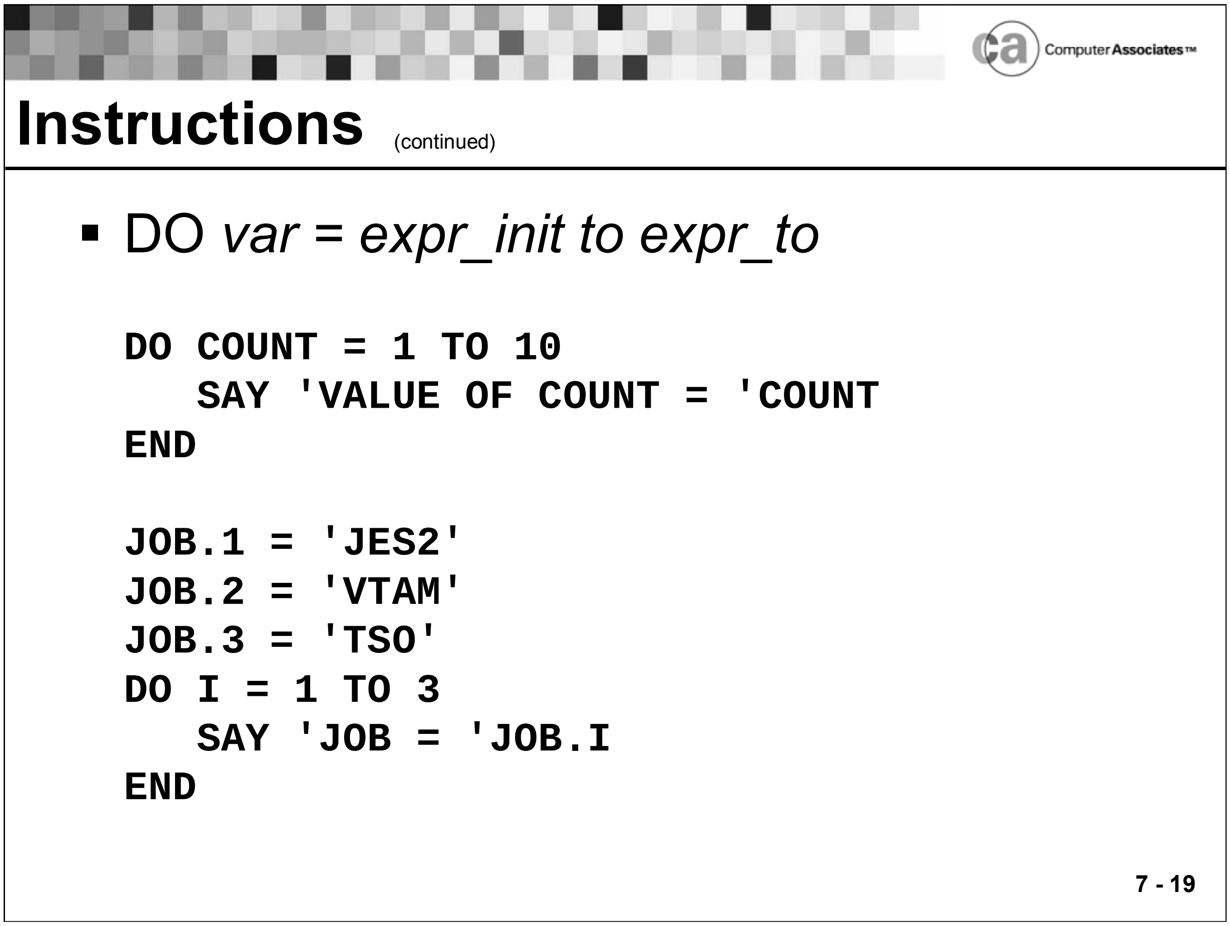
  ```
  DO expr_count ;
     [instructionlist]
  END ;
  ```

  expr_count - An expression which evaluates to a non negative whole number.

  instructionlist - A sequence of instructions.

  This example shows how you can execute an instruction five times:

  ```
  DO 5
    SAY 'REXX DO TEST'
  END
  ```



Instructions (continued)

- DO *var = expr_init to expr_to*

```
DO COUNT = 1 TO 10
   SAY 'VALUE OF COUNT = 'COUNT
END

JOB.1 = 'JES2'
JOB.2 = 'VTAM'
JOB.3 = 'TSO'
DO I = 1 TO 3
   SAY 'JOB = 'JOB.I
END
```

7 - 19

Types of Instructions

- DO var = expr_init to expr_to

  Use this statement to group instructions together and execute them repetitively (in a loop). It has the following form:

```
DO var = expr_init to expr_to ;
   [instructionlist]
END ;
```

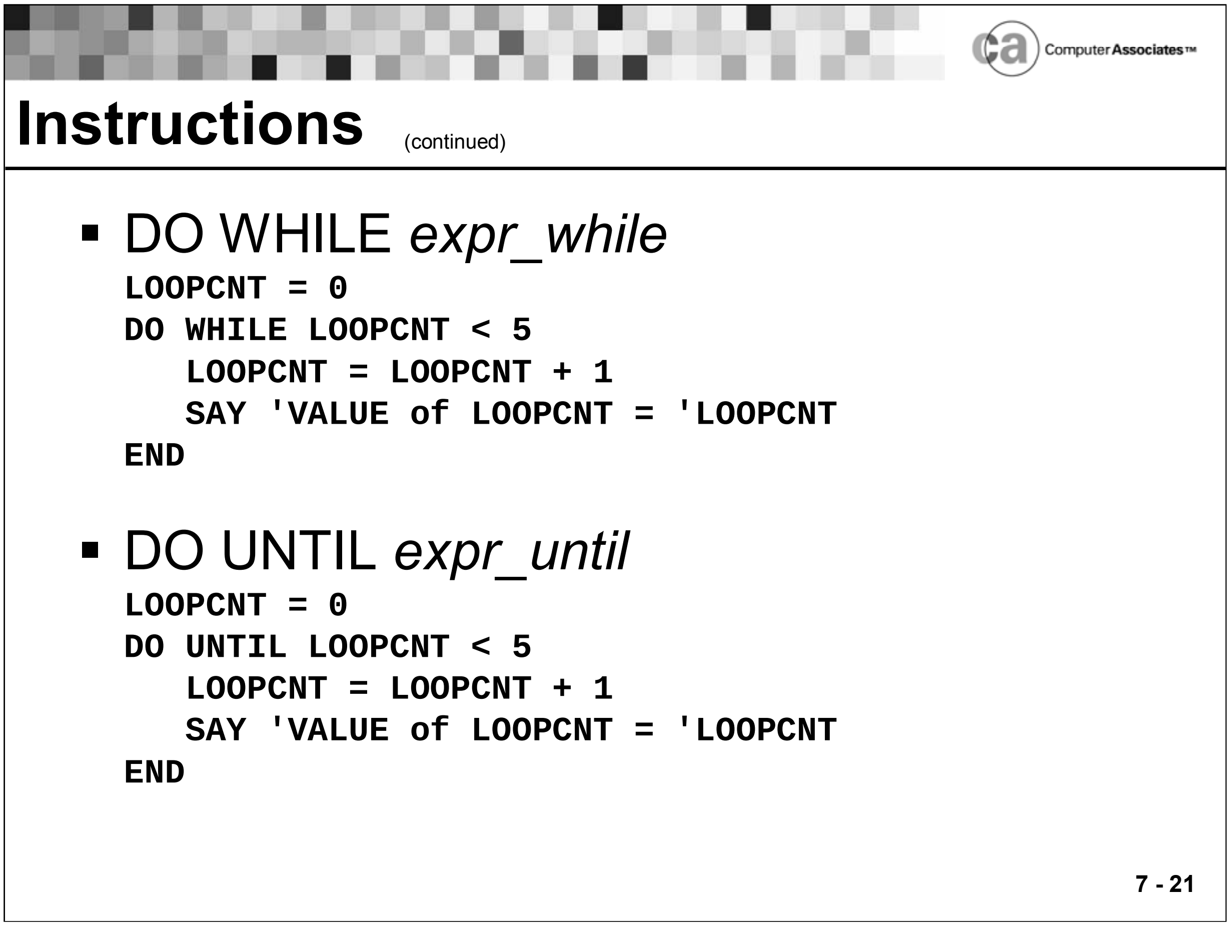
expr_init and expr_to - Expressions which evaluate to whole numbers of either sign.

Instructionlist - A sequence of instructions.

These examples show how you can execute a sequence of instructions more than once using a variable:

```
DO COUNT = 1 TO 10
    SAY 'VALUE OF COUNT = 'COUNT
END
```

```
JOB.1 = 'JES2'
JOB.2 = 'VTAM'
JOB.3 = 'TSO'
DO I = 1 TO 3
    SAY 'JOB = 'JOB.I
END
```



Instructions (continued)

- DO WHILE *expr_while*

```
LOOPCNT = 0
DO WHILE LOOPCNT < 5
   LOOPCNT = LOOPCNT + 1
   SAY 'VALUE of LOOPCNT = 'LOOPCNT
END
```

- DO UNTIL *expr_until*

```
LOOPCNT = 0
DO UNTIL LOOPCNT < 5
   LOOPCNT = LOOPCNT + 1
   SAY 'VALUE of LOOPCNT = 'LOOPCNT
END
```

7 - 21

Types of Instructions

- DO WHILE *expr_while*

  Use this statement to group instructions together and execute them conditionally. The WHILE test is made before each iteration. It has the following form:

```
DO WHILE expr_while ;
     [instructionlist]
END ;
```

expr_while - a logical expression which evaluates to zero or one.

instructionlist - A sequence of instructions.

Note: Make sure the value of the expression is false at some point so that the loop can be exited.

This example tests whether the value of the expression LOOPCNT is less than five:

```
LOOPCNT = 0
DO WHILE LOOPCNT < 5
     LOOPCNT = LOOPCNT + 1
     SAY 'VALUE of LOOPCNT = 'LOOPCNT
END
```

- DO UNTIL *expr_until*

  Use this statement to group instructions together and execute them conditionally. The UNTIL test is made after each iteration. It has the following form:

```
DO UNTIL expr_until ;
     [instructionlist]
END ;
```

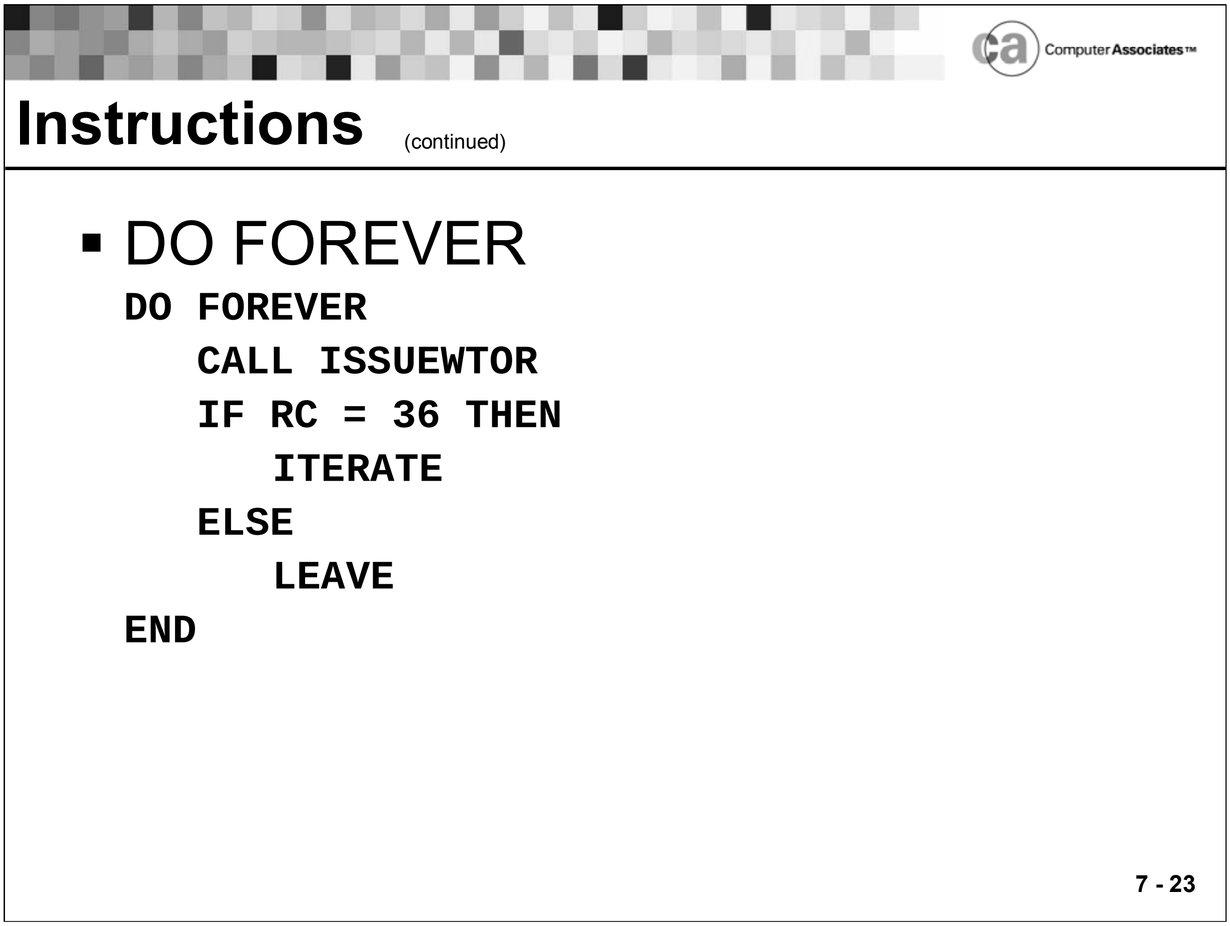
expr_until - a logical expression which evaluates to zero or one.

instructionlist - A sequence of instructions.

Note: Make sure the value of the expression is false at some point so that the loop can be exited.

This example tests whether the value of the expression LOOPCNT is less than five:

```
LOOPCNT = 0
DO UNTIL LOOPCNT < 5
     LOOPCNT = LOOPCNT + 1
     SAY 'VALUE of LOOPCNT = 'LOOPCNT
END
```



Instructions (continued)

- DO FOREVER

```
DO FOREVER
   CALL ISSUEWTOR
   IF RC = 36 THEN
      ITERATE
   ELSE
      LEAVE
END
```

7 - 23

Types of Instructions

- DO FOREVER

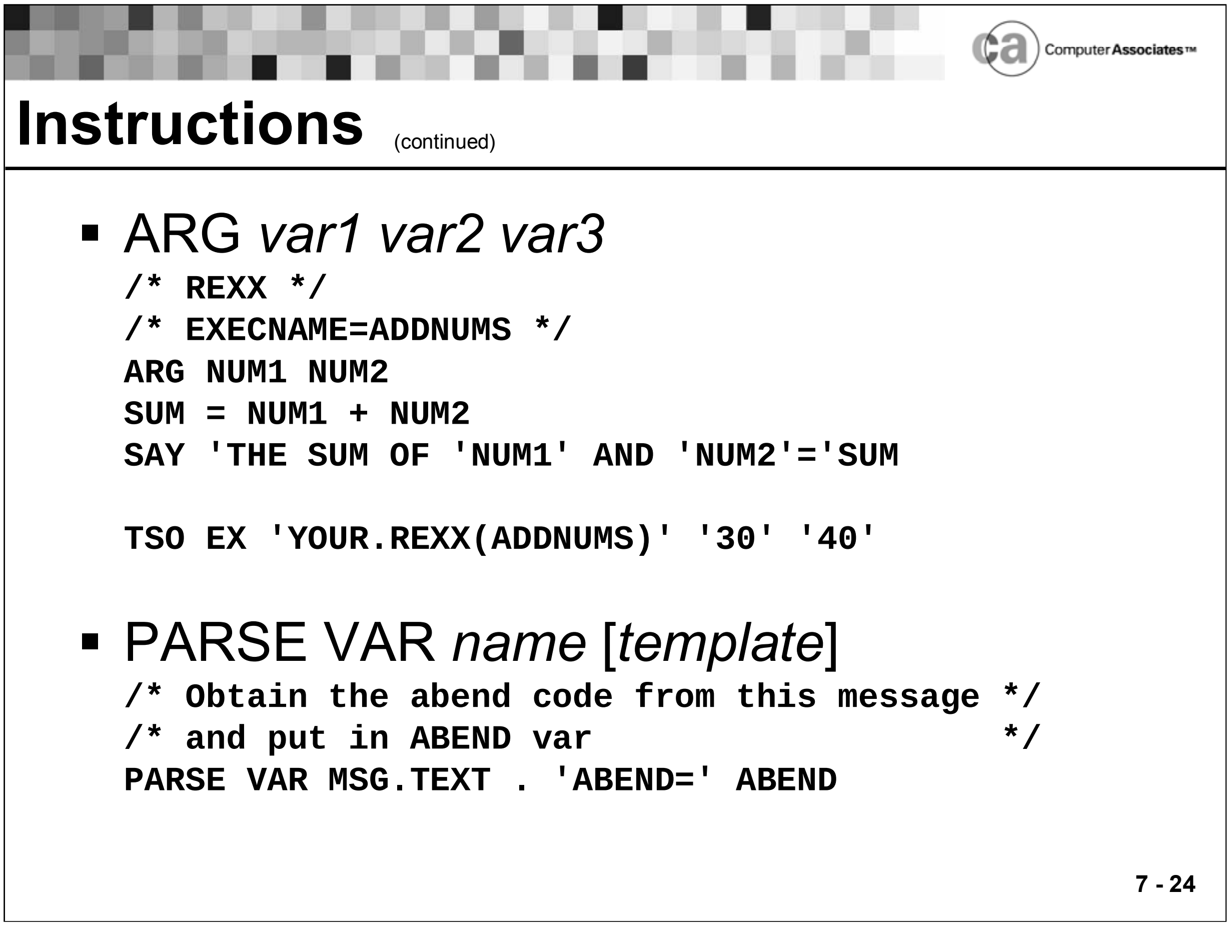
Use this statement to group instructions together and execute them forever (until the condition is satisfied). It has the following form:

```
DO FOREVER ;
     [instructionlist]
END ;
```

instructionlist - A sequence of instructions

The following rule logic executes forever unless the ISSUEWTOR subroutine sets a return code of 36:

```
DO FOREVER
  CALL ISSUEWTOR
  IF RC = 36 THEN ITERATE
       ELSE LEAVE
END
```



Instructions (continued)

- ARG *var1 var2 var3*

```
/* REXX */
/* EXECNAME=ADDNUMS */
ARG NUM1 NUM2
SUM = NUM1 + NUM2
SAY 'THE SUM OF 'NUM1' AND 'NUM2'='SUM

TSO EX 'YOUR.REXX(ADDNUMS)' '30' '40'
```

- PARSE VAR *name* [*template*]

```
/* Obtain the abend code from this message */
/* and put in ABEND var                    */
PARSE VAR MSG.TEXT . 'ABEND=' ABEND
```

7 - 24

Types of Instructions

- ARG var1 var2 var3

Use this statement to enable your REXX EXEC to be invoked with arguments. It has the following form:

```
ARG [variable] ;
```

variable - A list of symbols separated by blanks and/or patterns.

When the following REXX code is invoked via the TSO EX statement, it causes the program to execute the arguments NUM1=30 and NUM2=40:

```
/* REXX */
/* EXECNAME=ADDNUMS */
ARG NUM1 NUM2
SUM = NUM1 + NUM2
SAY 'THE SUM OF 'NUM1' AND 'NUM2'='SUM
```

To execute the code, enter the following:

```
TSO EX 'YOUR.REXX(ADDNUMS)' '30' '40'
```

- PARSE VAR name [template]

  Use this statement to assign data to one or more variables. It has the following form:

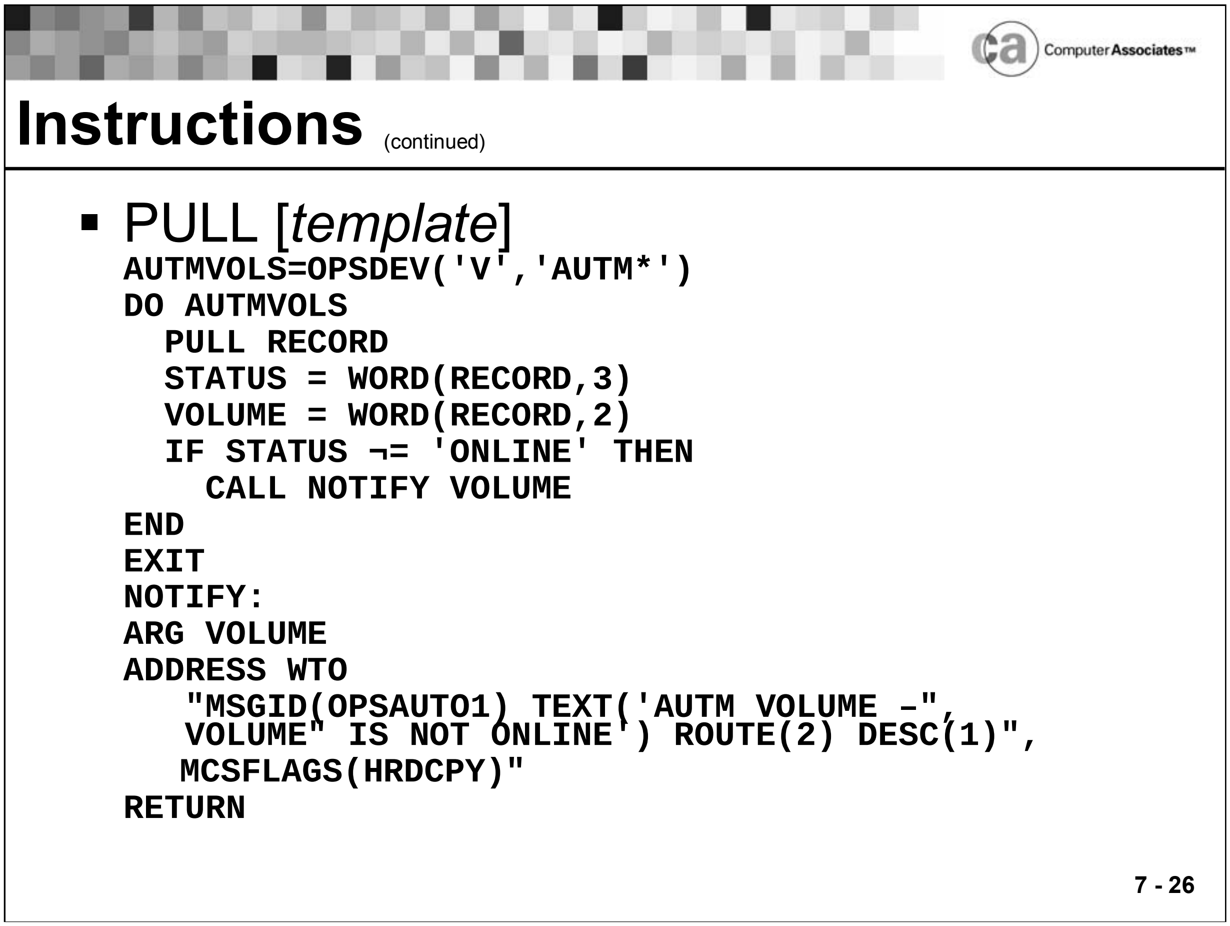
  ```
  PARSE VAR name [template] ;
  ```

  name - A valid variable name (that is, it cannot begin with a period or digit). The value of the variable specified by name is parsed.

  template - A list of symbols separated by blanks and/or patterns.

  This example illustrates breaking down a data string:

  ```
  /* Obtain the abend code from this message */
  /* and put in ABEND var                    */
  PARSE VAR MSG.TEXT . 'ABEND=' ABEND
  ```



Instructions (continued)

- PULL [*template*]

```
AUTMVOLS=OPSDEV('V','AUTM*')
DO AUTMVOLS
  PULL RECORD
  STATUS = WORD(RECORD,3)
  VOLUME = WORD(RECORD,2)
  IF STATUS ¬= 'ONLINE' THEN
    CALL NOTIFY VOLUME
END
EXIT
NOTIFY:
ARG VOLUME
ADDRESS WTO
   "MSGID(OPSAUTO1) TEXT('AUTM VOLUME -",
   VOLUME" IS NOT ONLINE') ROUTE(2) DESC(1)",
   MCSFLAGS(HRDCPY)"
RETURN
```

7 - 26

Types of Instructions

- PULL [template]

  Use this statement to retrieve one record from the external data queue (EDQ). You can think of the EDQ as an internal “scratch pad,” or stack, to which various OPS/REXX host environments and functions return requested data when they are invoked via an AOF rule or OPS/REXX program that is running in an OPSOSF server.

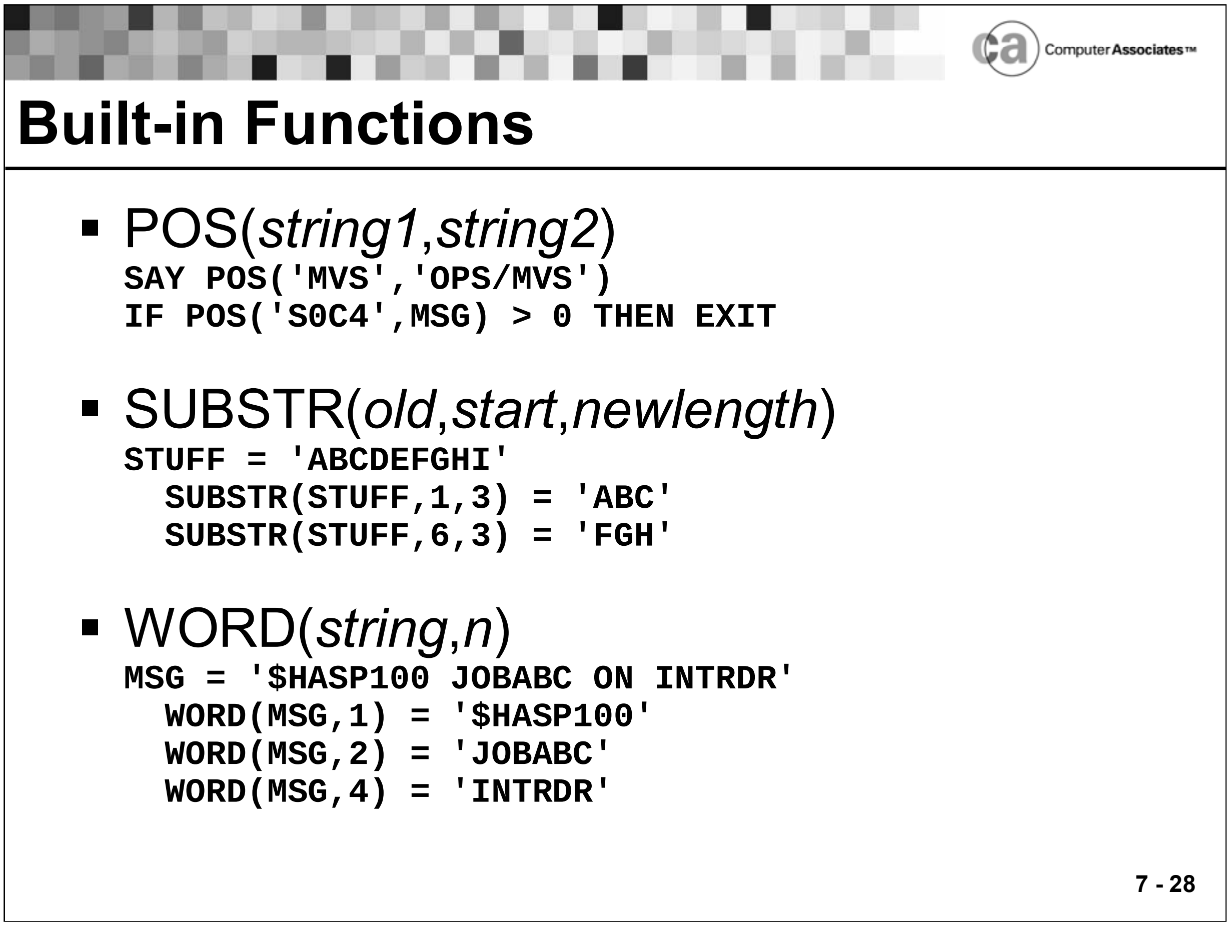
  The PULL instruction has the following form:

  ```
  PULL [template] ;
  ```

  template - A list of symbols separated by blanks and/or patterns.

Example:

```
AUTMVOLS=OPSDEV('V','AUTM*')
DO AUTMVOLS
  PULL RECORD
  STATUS = WORD(RECORD,3)
  VOLUME = WORD(RECORD,2)
  IF STATUS ¬= 'ONLINE' THEN
    CALL NOTIFY VOLUME
END
EXIT
NOTIFY:
ARG VOLUME
ADDRESS WTO
   "MSGID(OPSAUTO1) TEXT('AUTM VOLUME -",
   VOLUME" IS NOT ONLINE') ROUTE(2) DESC(1)",
   MCSFLAGS(HRDCPY)"
RETURN
```



Built-in Functions

- POS(*string1*,*string2*)

```
SAY POS('MVS','OPS/MVS')
IF POS('S0C4',MSG) > 0 THEN EXIT
```

- SUBSTR(*old*,*start*,*newlength*)

```
STUFF = 'ABCDEFGHI'
  SUBSTR(STUFF,1,3) = 'ABC'
  SUBSTR(STUFF,6,3) = 'FGH'
```

- WORD(*string*,*n*)

```
MSG = '$HASP100 JOBABC ON INTRDR'
  WORD(MSG,1) = '$HASP100'
  WORD(MSG,2) = 'JOBABC'
  WORD(MSG,4) = 'INTRDR'
```

7 - 28

Built-in Functions

- In REXX, “built-in functions” are special reserved routines that can receive data, process it, and return a value. As you will learn in an upcoming lesson, OPS/REXX has its own set of built-in functions that provide you with even greater functionality.

Types of Built-in Functions

- POS(string1,string2)

  Use this function to return the position of one string, string1 in another, string2. If the string is not found, 0 is returned.

  Example:

```
SAY POS('MVS','OPS/MVS')
IF POS('S0C4',MSG) > 0 THEN EXIT
```

- SUBSTR(old,start,newlength)

  Use this function to return a portion of old, beginning with start position and continuing for newlength.

Example:

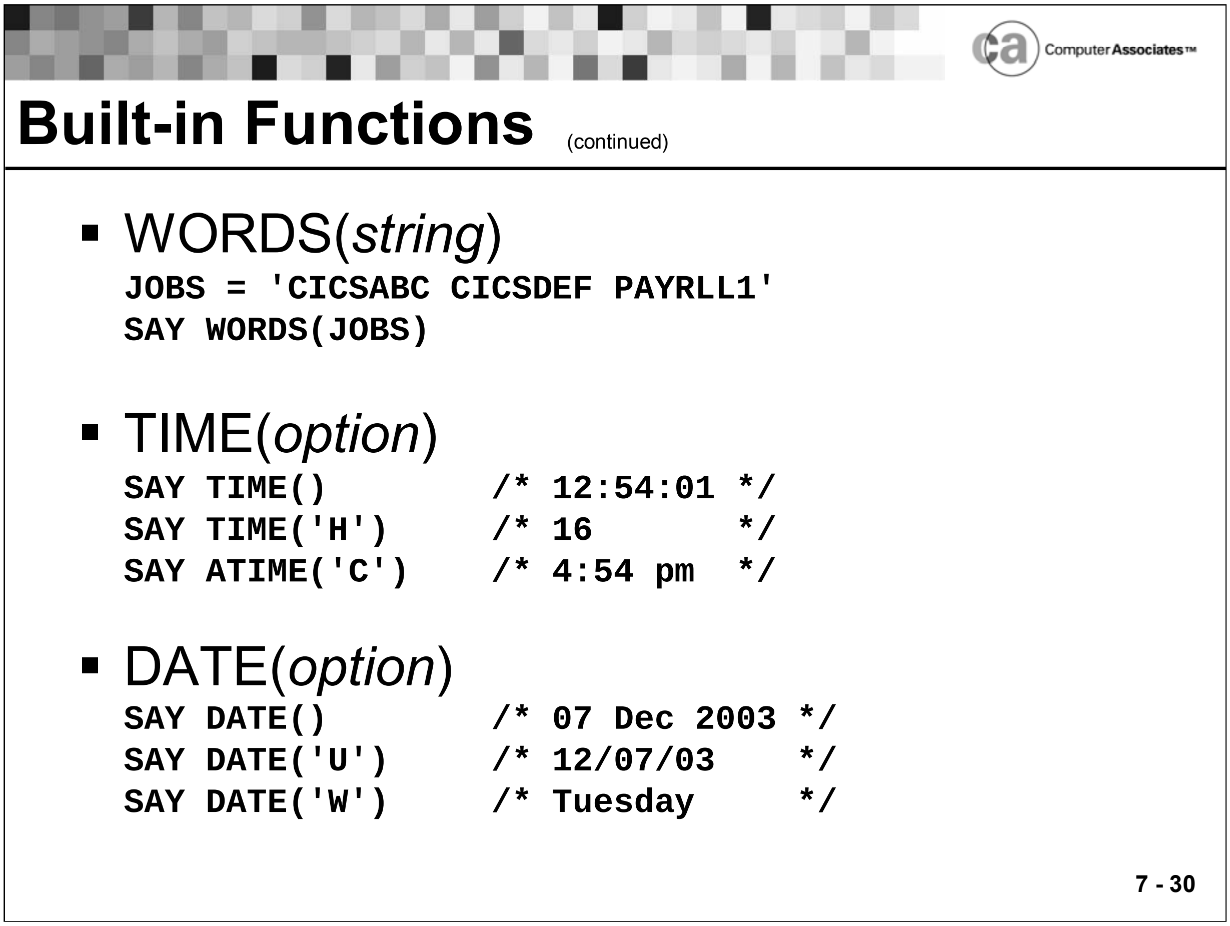
```
STUFF = 'ABCDEFGHI'
  SUBSTR(STUFF,1,3) = 'ABC'
  SUBSTR(STUFF,6,3) = 'FGH'
```

- WORD(string,n)

  Use this function to return the nth blank-delimited word in string.

  The following example illustrates using the WORD() function against the variable MSG:

```
MSG = '$HASP100 JOBABC ON INTRDR’
  WORD(MSG,1) = '$HASP100'
  WORD(MSG,2) = 'JOBABC'
  WORD(MSG,4) = 'INTRDR'
```



Built-in Functions (continued)

- WORDS(*string*)

```
JOBS = 'CICSABC CICSDEF PAYRLL1'
SAY WORDS(JOBS)
```

- TIME(*option*)

```
SAY TIME()        /* 12:54:01 */
SAY TIME('H')     /* 16       */
SAY ATIME('C')    /* 4:54 pm  */
```

- DATE(*option*)

```
SAY DATE()        /* 07 Dec 2003 */
SAY DATE('U')     /* 12/07/03    */
SAY DATE('W')     /* Tuesday     */
```

7 - 30

Types of Functions

- WORDS(string)

  Use this function to return the number of blank-delimited words in string.

  Example:

  ```
  JOBS = 'CICSABC CICSDEF PAYRLL1'
  SAY WORDS(JOBS)
  ```

- TIME(option)

  Use this function to return the system time in the format specified by option.

  This example returns the time in the default format (hh:mm:ss):

  ```
  SAY TIME()        /* 12:54:01 */
  ```

  This example returns the time in the hours format (hh):

  ```
  SAY TIME('H')     /* 16       */
  ```

  This example returns the time in the civil format (hh:mmxx):

  ```
  SAY ATIME('C')    /* 4:54 pm  */
  ```

- DATE(option)

  Use this function to return the system date in the format specified by option.

  This example returns the date in the default format (dd Mmm yyyy):

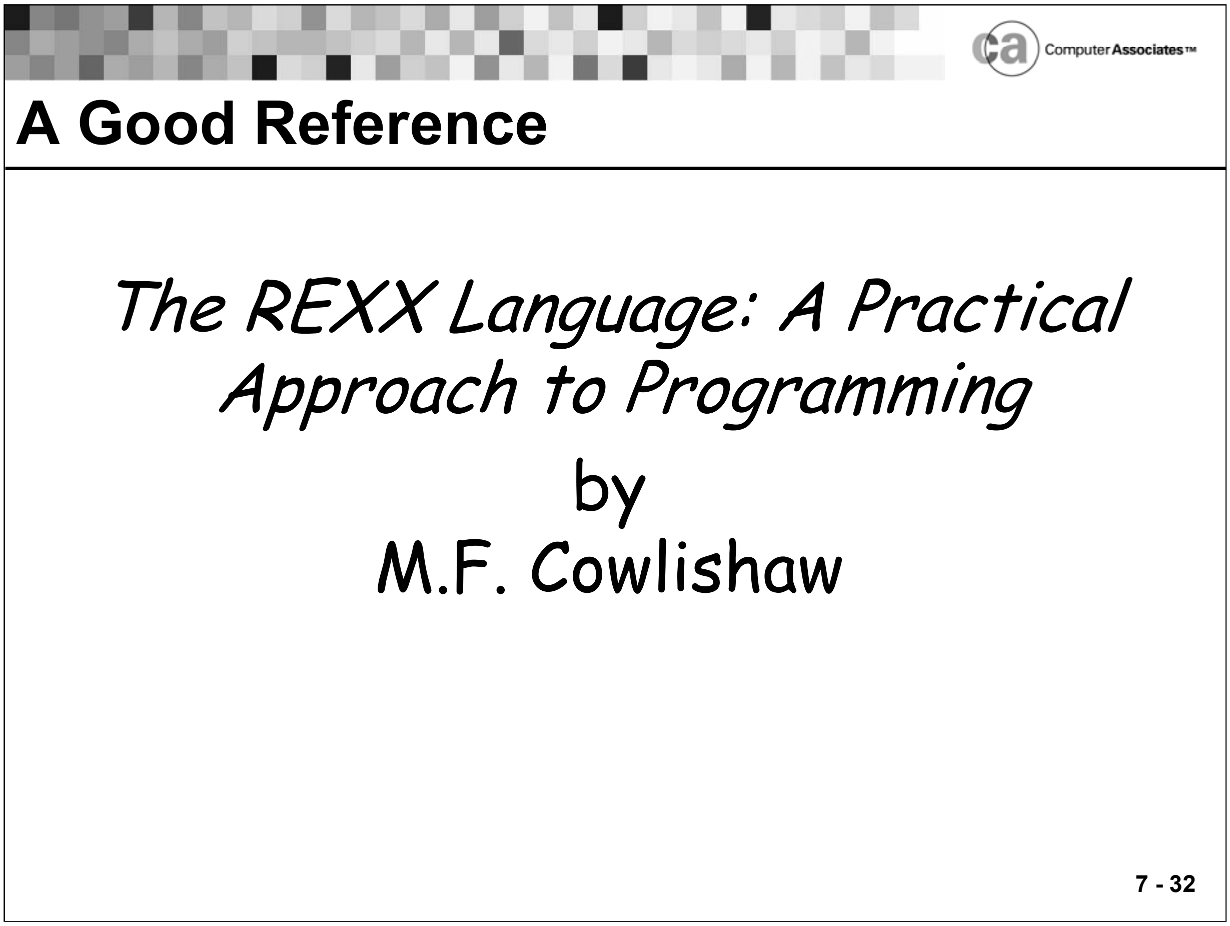
  ```
  SAY DATE()        /* 07 Dec 2003 */
  ```

  This example returns the date in the USA format (mm/dd/yy):

  ```
  SAY DATE('U')     /* 12/07/03    */
  ```

  This example returns the date in the weekday format:

  ```
  SAY DATE('W')     /* Tuesday     */
  ```



A Good Reference

The REXX Language: A Practical Approach to Programming

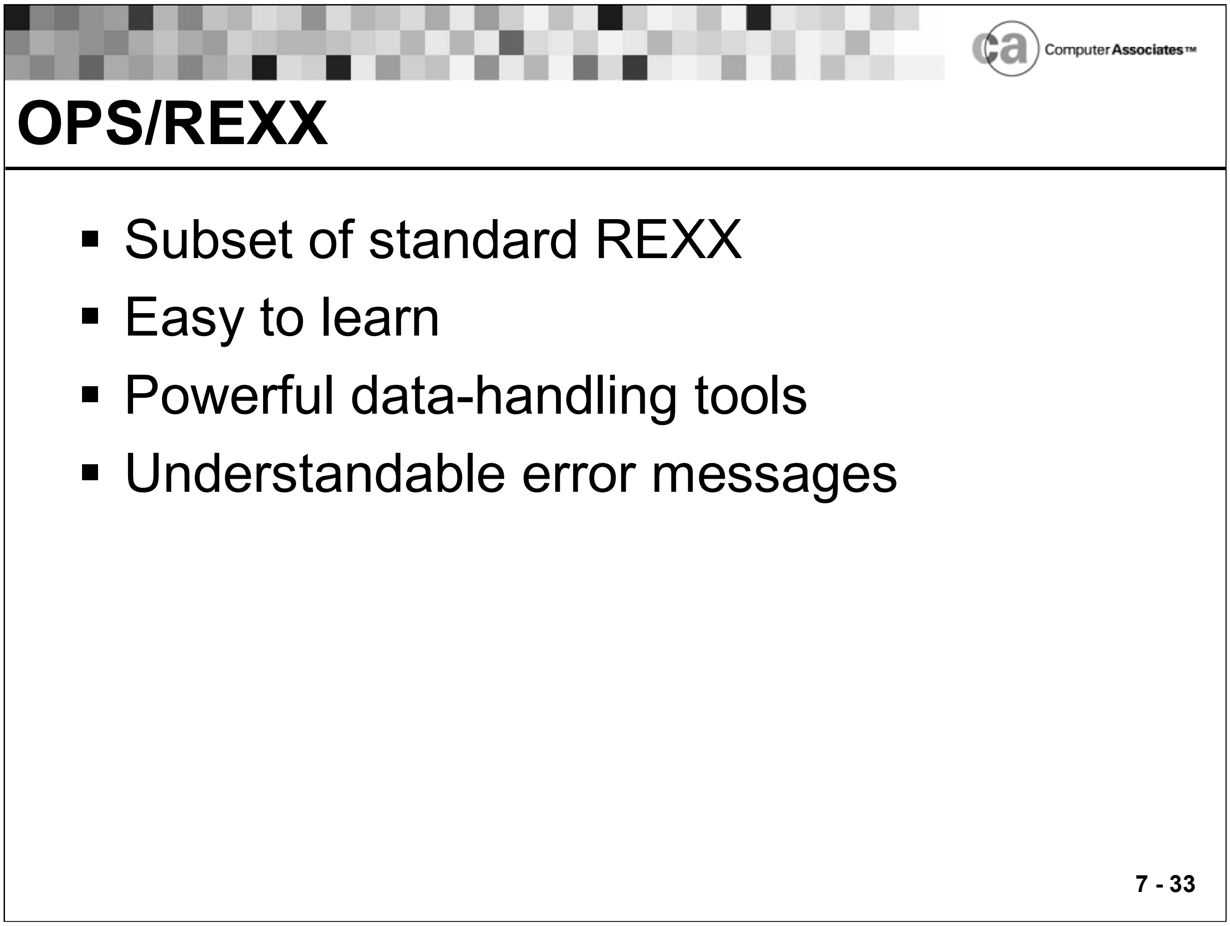
by

M.F. Cowlishaw

7 - 32

A Good Reference

For more detailed information about standard REXX, refer to any REXX manual. A highly recommended book is THE REXX LANGUAGE: A Practical Approach to Programming by M.F. Cowlishaw. You can order this book from Prentice-Hall.



OPS/REXX

- Subset of standard REXX
- Easy to learn
- Powerful data-handling tools
- Understandable error messages

7 - 33

OPS/REXX

- A crucial part of Unicenter CA-OPS/MVS, 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 OS/390 operations.
- Because OPS/REXX differs only slightly from standard REXX, this section explains the differences between OPS/REXX and standard REXX.
- CA chose REXX as the programming language for 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 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 OS/390. Approaches that try to use standard 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, CA has adopted these well thought-out error messages for Unicenter CA-OPS/MVS. The Cowlishaw book describes messages with error codes up to 49. Online message documentation describes errors with higher codes.

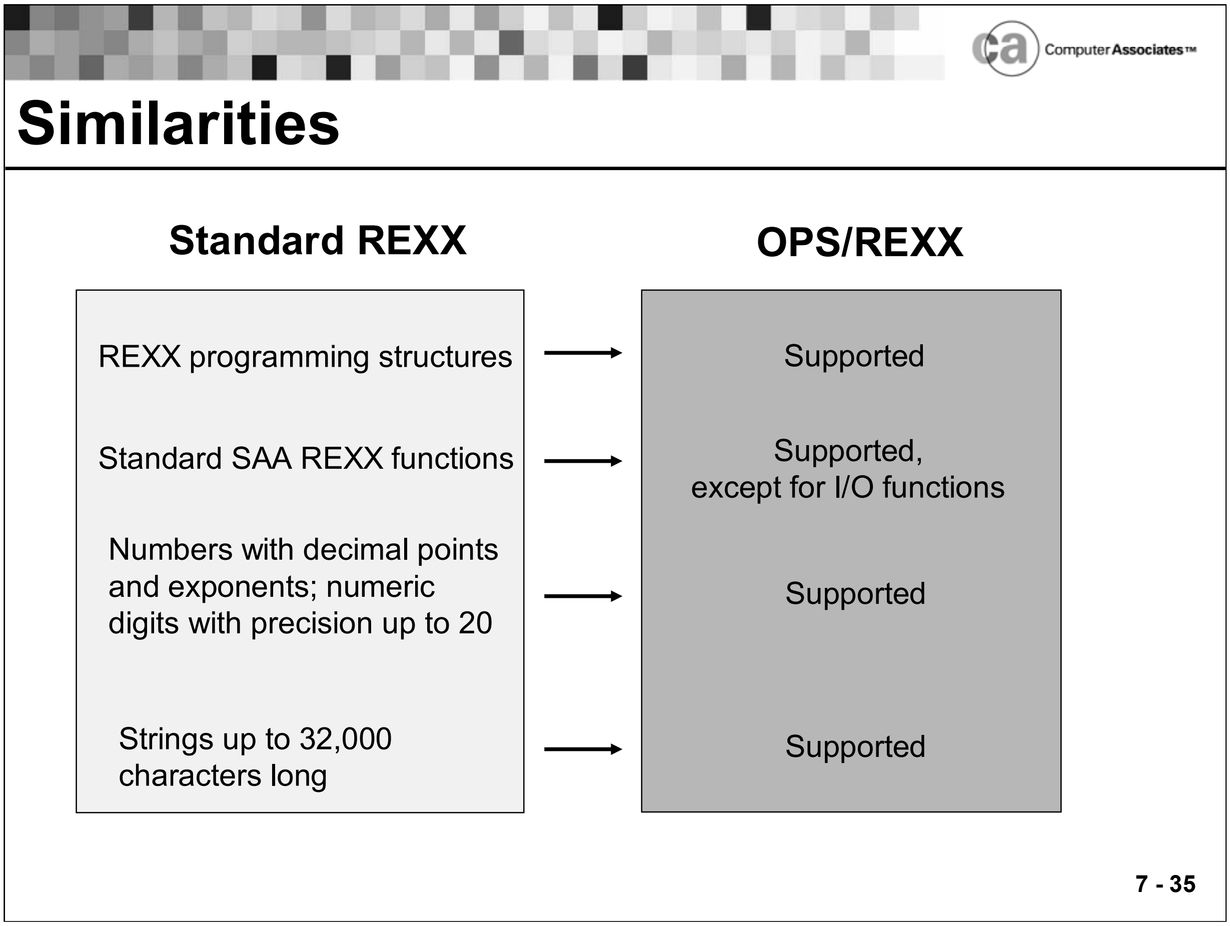
Uses of OPS/REXX Within Unicenter CA-OPS/MVS

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

  AOF rules are actually OPS/REXX programs that can respond automatically to system events. The availability of OPS/REXX's 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. OPS/REXX's interface to the Dialog Manager is as complete and powerful as that of standard REXX.

  You can write AOF asynchronous procedures (that execute in Unicenter CA-OPS/MVS server address spaces) in both OPS/REXX and the standard REXX language.



Similarities

Standard REXX	OPS/REXX
REXX programming structures	Supported
Standard SAA REXX functions	Supported, except for I/O functions
Numbers with decimal points and exponents; numeric digits with precision up to 20	Supported
Strings up to 32,000 characters long	Supported

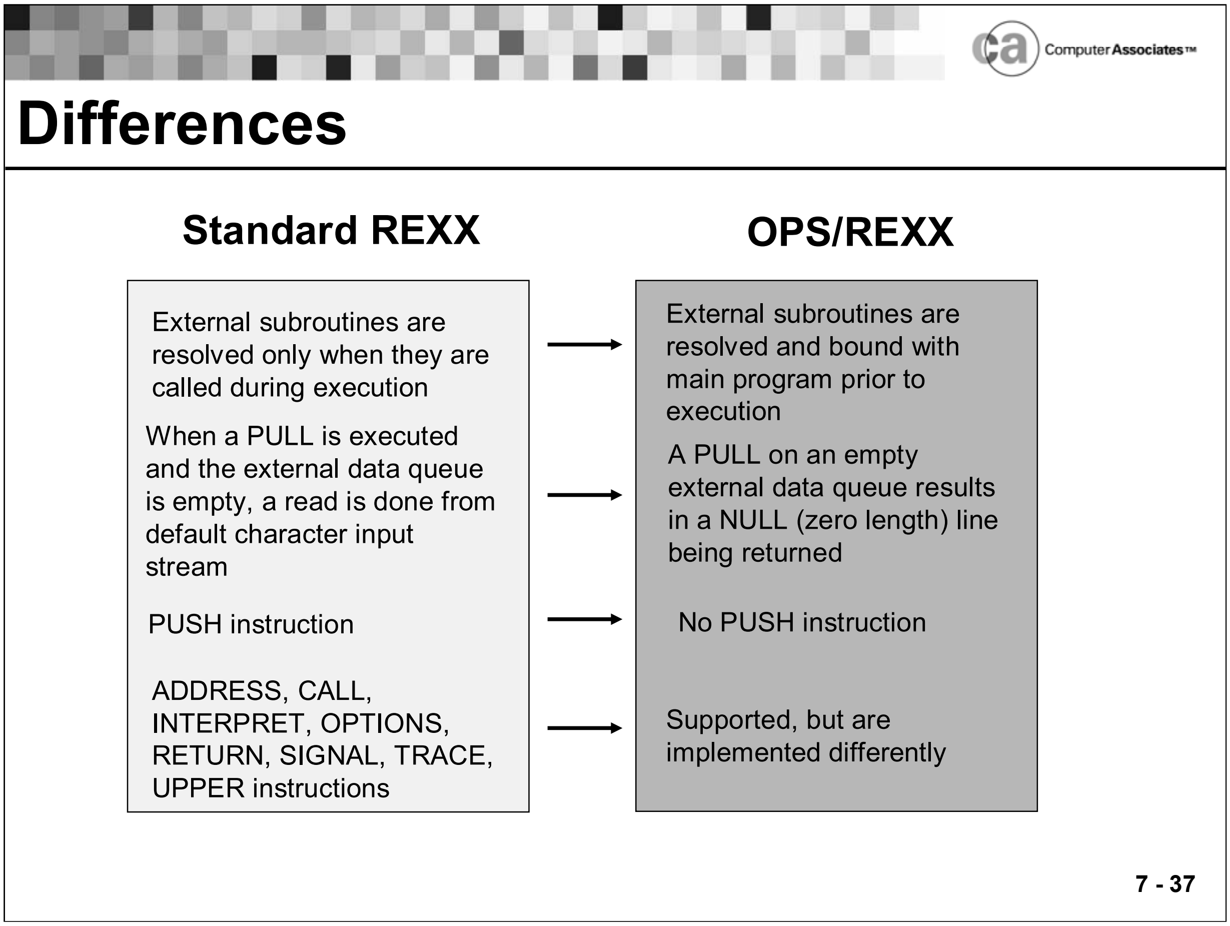
Similarities

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.

Unicenter CA-OPS/MVS REXXMAXSTRINGLENGTH parameter enables you to use a lower maximum string length if you wish.



Differences

Standard REXX		OPS/REXX
External subroutines are resolved only when they are called during execution	→	External subroutines are resolved and bound with main program prior to execution
When a PULL is executed and the external data queue is empty, a read is done from default character input stream	→	A PULL on an empty external data queue results in a NULL (zero length) line being returned
PUSH instruction	→	No PUSH instruction
ADDRESS, CALL, INTERPRET, OPTIONS, RETURN, SIGNAL, TRACE, UPPER instructions	→	Supported, but are implemented differently

7 - 37

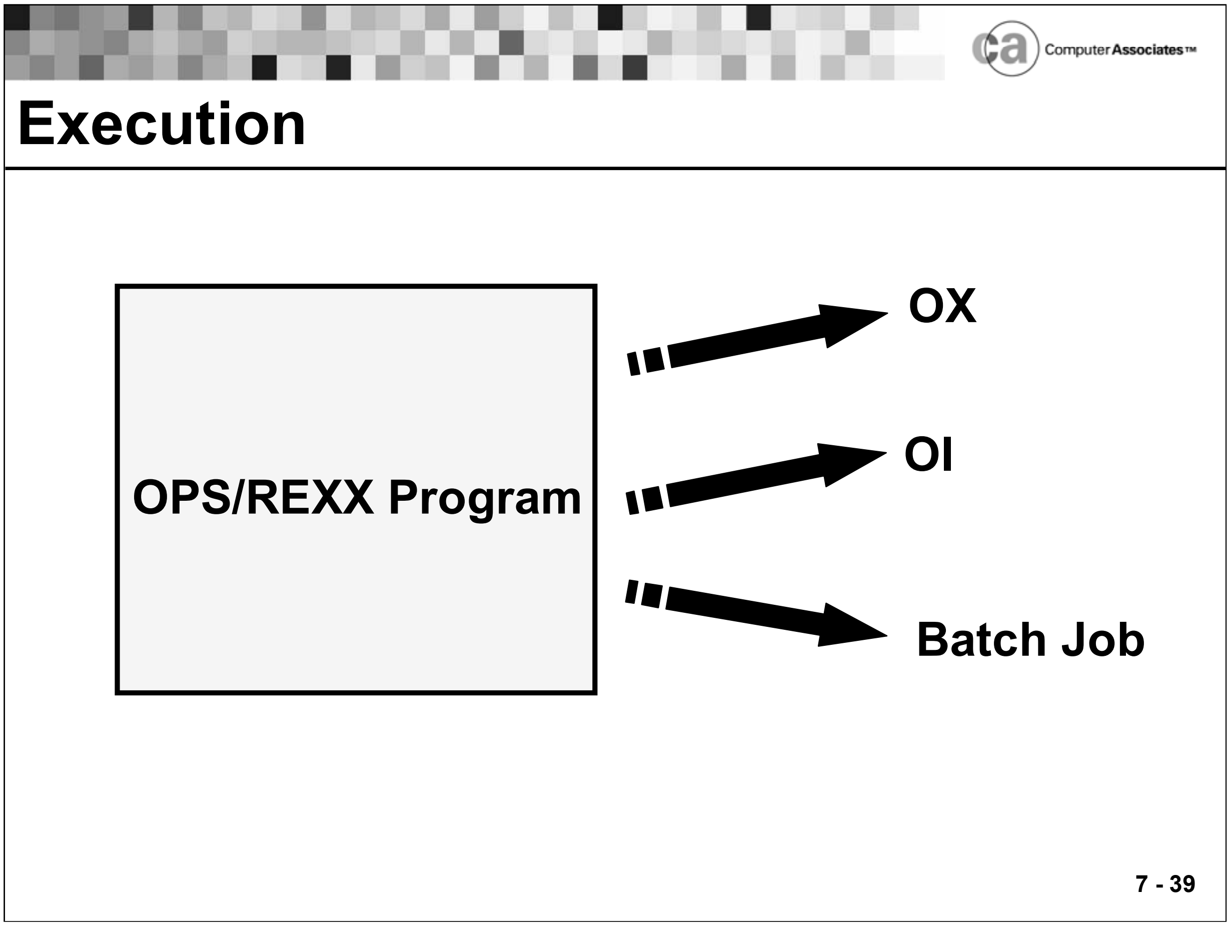
Differences

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 standard REXX, external subroutines are resolved only when they are called during execution.
- In Unicenter CA-OPS/MVS version 4.1.0 or higher, 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 standard REXX, so the portability of REXX code that uses this instruction is unaffected.
- 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 within a rule, the only possible default character input stream is the console. Prompting the operator for the next line of input would be undesirable. 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.
- The ADDRESS, CALL, INTERPRET, OPTIONS, RETURN, SIGNAL, TRACE, and UPPER instructions are supported in OPS/REXX, but they are implemented differently than they are in standard REXX. See your Unicenter CA-OPS/MVS documentation for specific details.



Execution

OPS/REXX Program

OX

OI

Batch Job

7 - 39

Execution

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

- Explicitly, via the OX (OPSEXEC) command.
- Implicitly, via the OI (OPSIMEX) command.
- As batch jobs.

Use the OX (OPSEXEC) and OI (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.

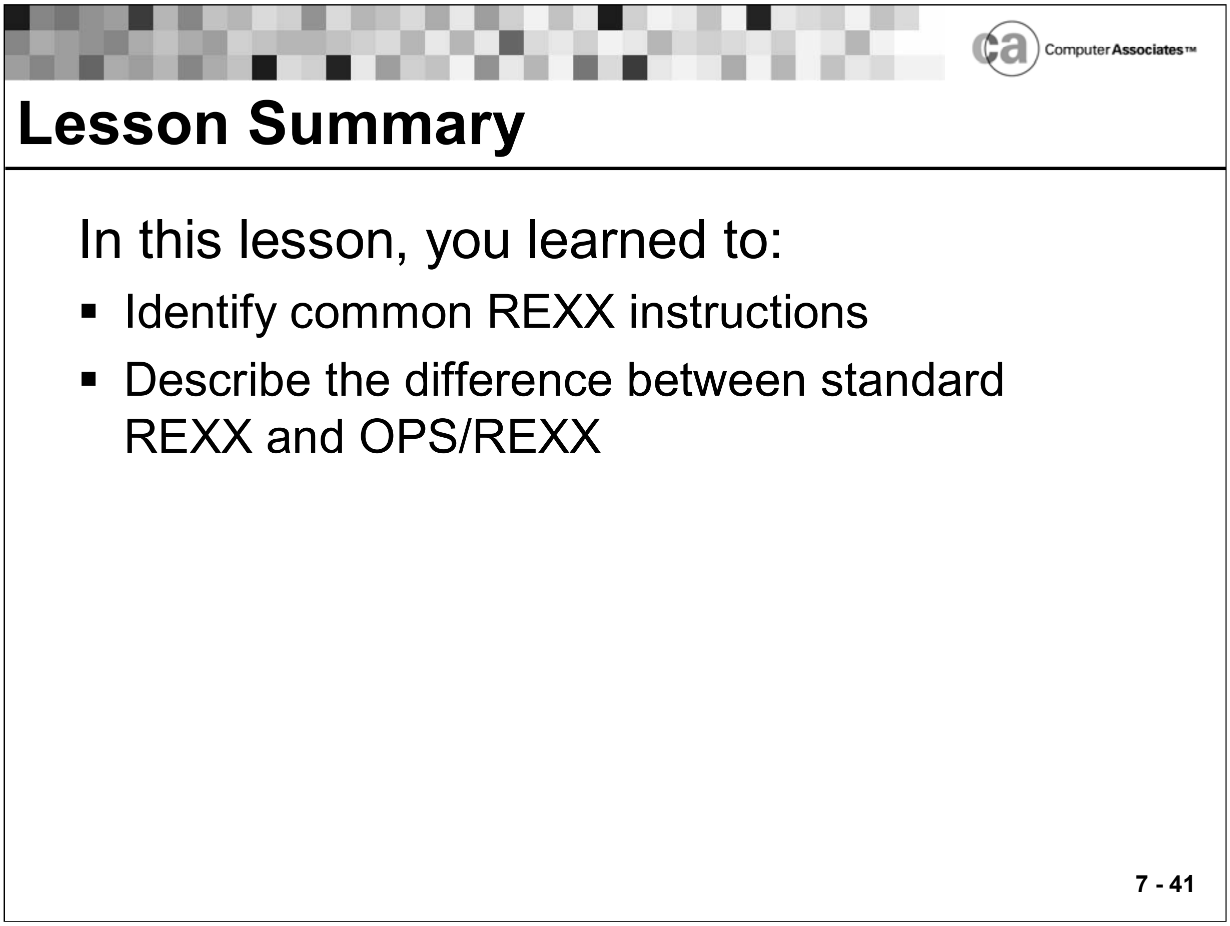
When used with the AOF, OPS/REXX programs have a special structure and are called rules. Outside the AOF environment, OPS/REXX programs are simply called programs.

Unless you have precompiled rules, Unicenter CA-OPS/MVS 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 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:

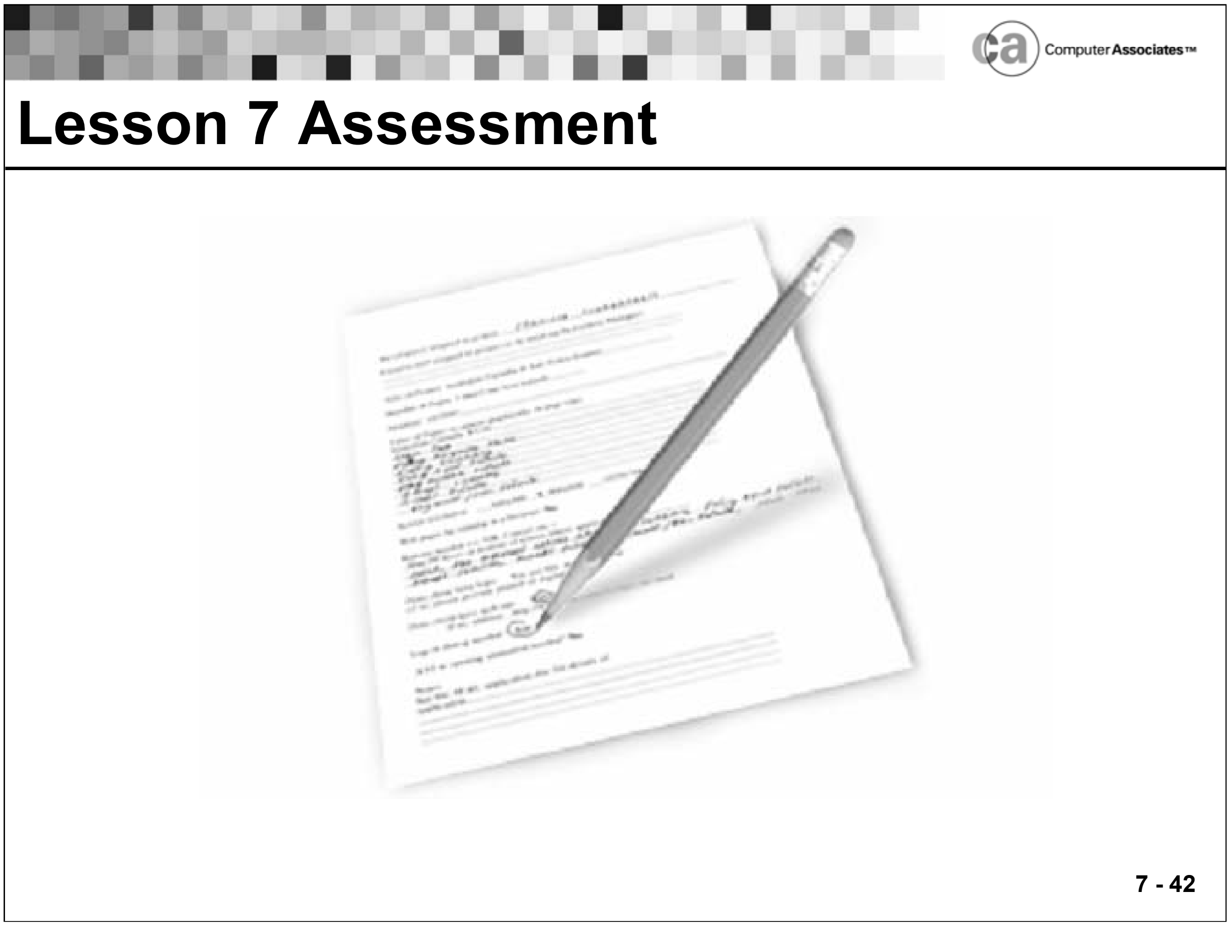
- With explicit execution, you supply the name of the data set that contains the program.
- With implicit execution, you provide only a program name. OPS/REXX then locates that program in the library or libraries allocated to DDNAME OPSEXEC or SYSEXEC.



Lesson Summary

In this lesson, you learned to:

- Identify common REXX instructions
- Describe the difference between standard REXX and OPS/REXX



Lesson 7 Assessment

7 - 42

Lesson 7 Assessment

1. True/False: OPS/REXX adds to standard REXX a set of extensions that automate and enhance the productivity of OS/390 operations.

____

2. What would you use this REXX instruction for? PARSE VAR name [template]

____

3. Which REXX function returns the time in this format? /* 11:56:04 */

____

4. Which REXX instruction is used to group instructions together and execute them conditionally?

____

Lesson 7 Assessment (continued)

5. True/False: REXX programs are built upon clauses.

____

6. The PUSH instruction is not available in OPS/REXX. Which alternative instruction can you use to achieve the same results?

____

7. What would you use this REXX instruction for? IF…THEN…ELSE

____

8. Which REXX operator combines two strings by appending the second string to the end of the first one?

____

9. Name four REXX arithmetic operators.

____

10. True/False: OPS/REXX programs are called rules outside the AOF environment.

____

11. True/False: Standard SAA REXX I/O functions are supported in OPS/REXX.

____

12. What would this function return? SAY DATE('U')

____

13. True/False: In OPS/REXX, a PULL instruction to an empty external data queue results in a null line being returned.

____

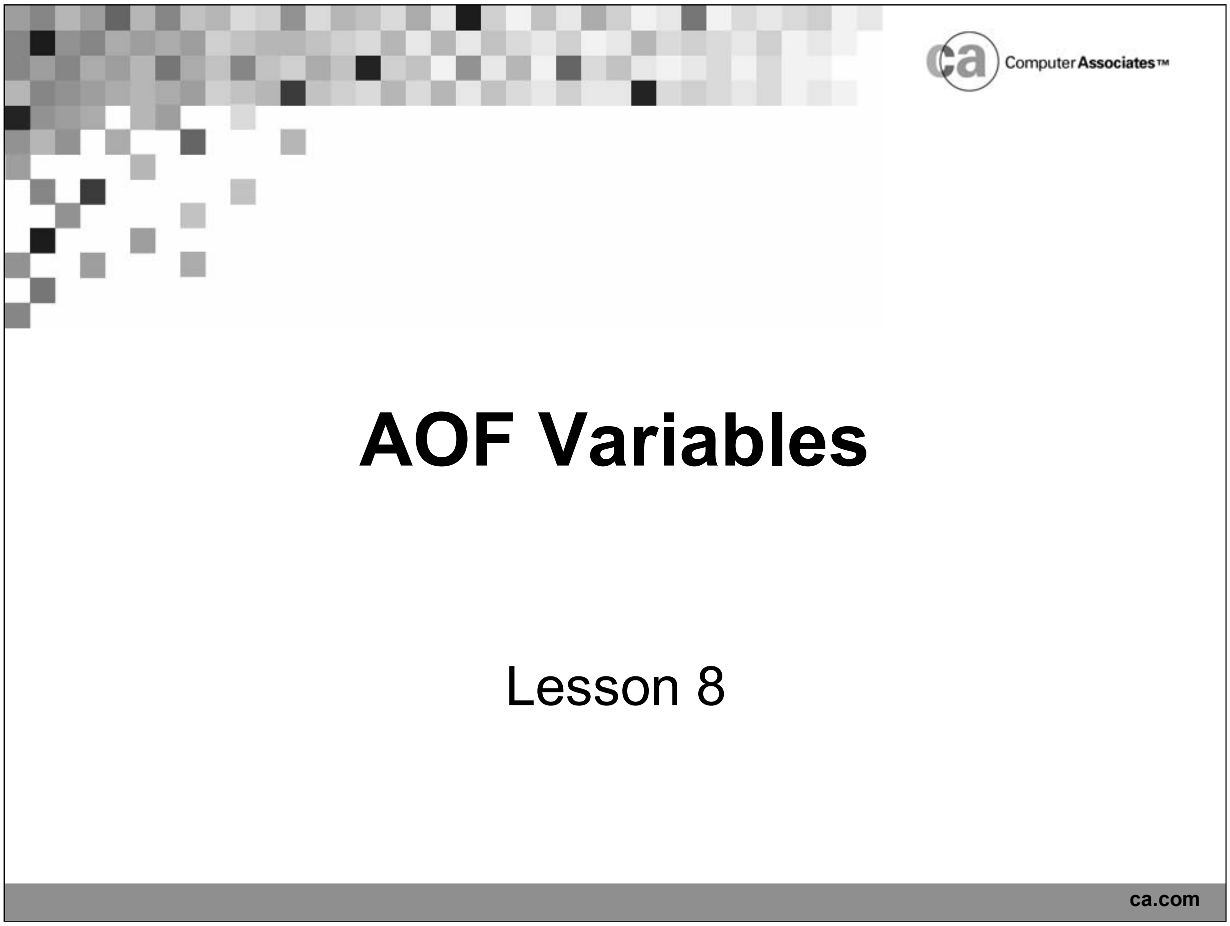
Lesson 7 Assessment (continued)

14. Define an instruction in REXX.

    ____

15. Describe the difference in how standard REXX and OPS/REXX resolve external subroutines.

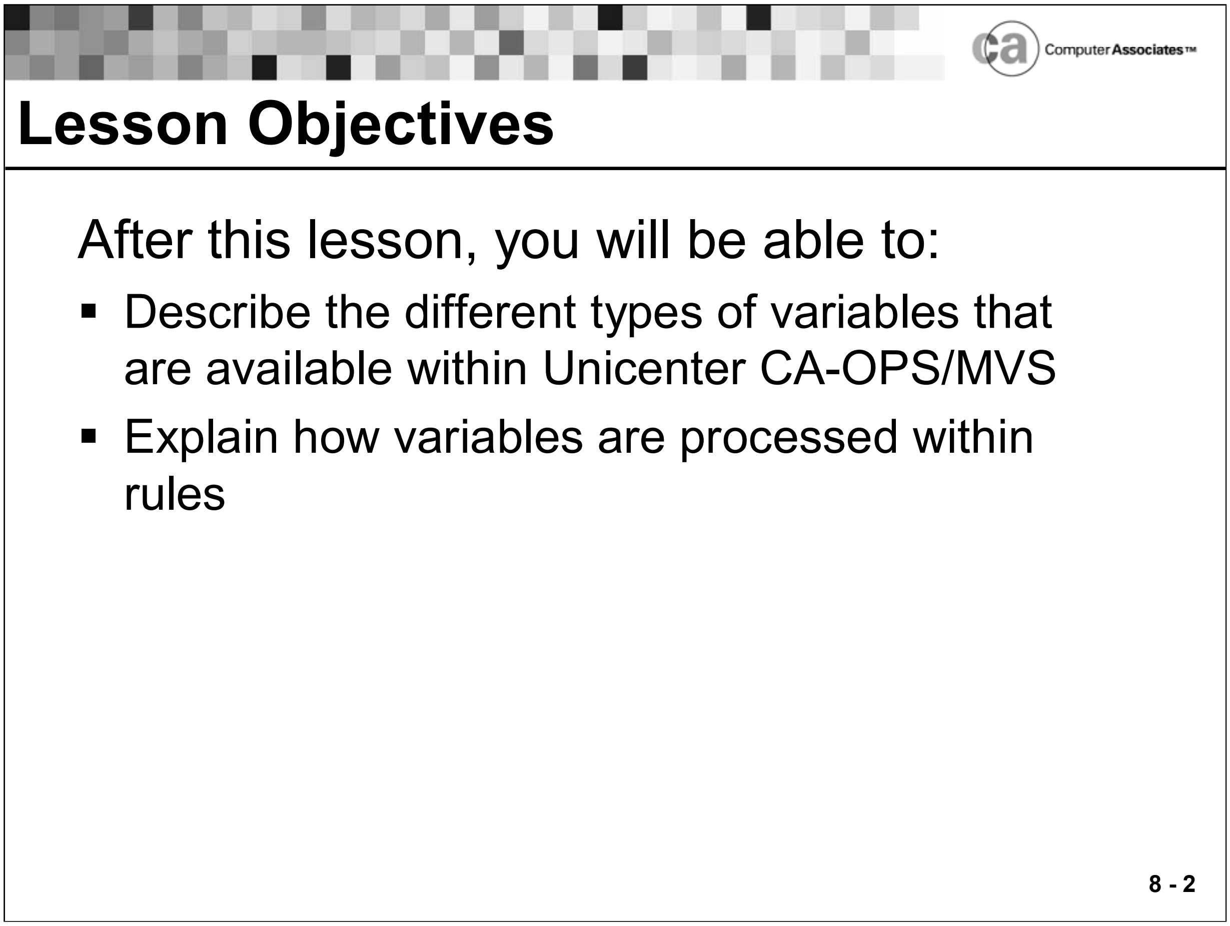
    ____



AOF Variables

Lesson 8

ca.com

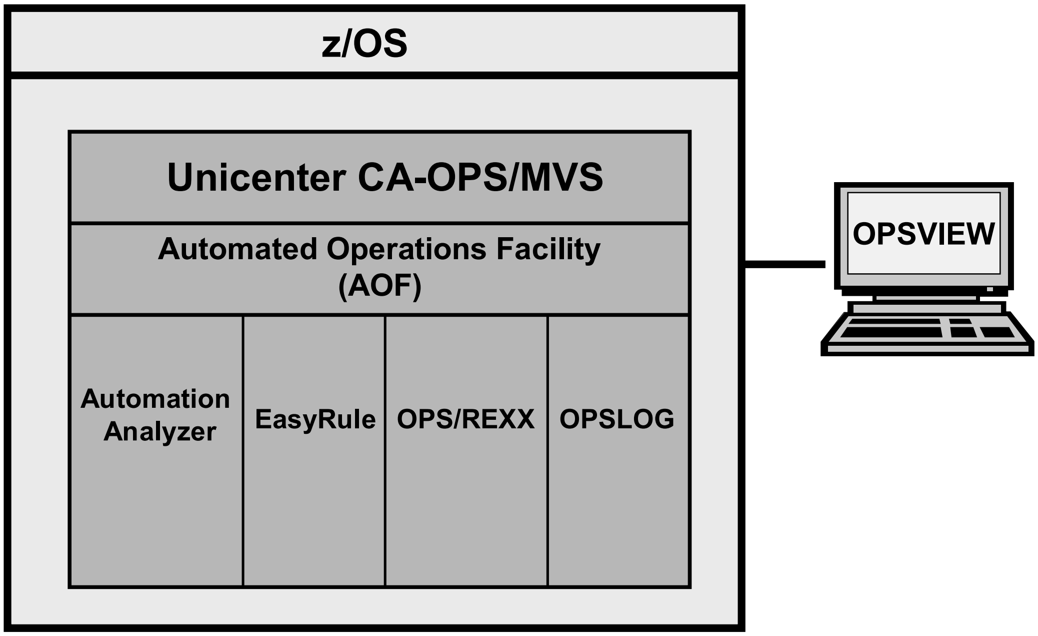


Lesson Objectives

After this lesson, you will be able to:

- Describe the different types of variables that are available within Unicenter CA-OPS/MVS
- Explain how variables are processed within rules

AOF Variables



z/OS

Unicenter CA-OPS/MVS

Automated Operations Facility
(AOF)

Automation Analyzer

EasyRule

OPS/REXX

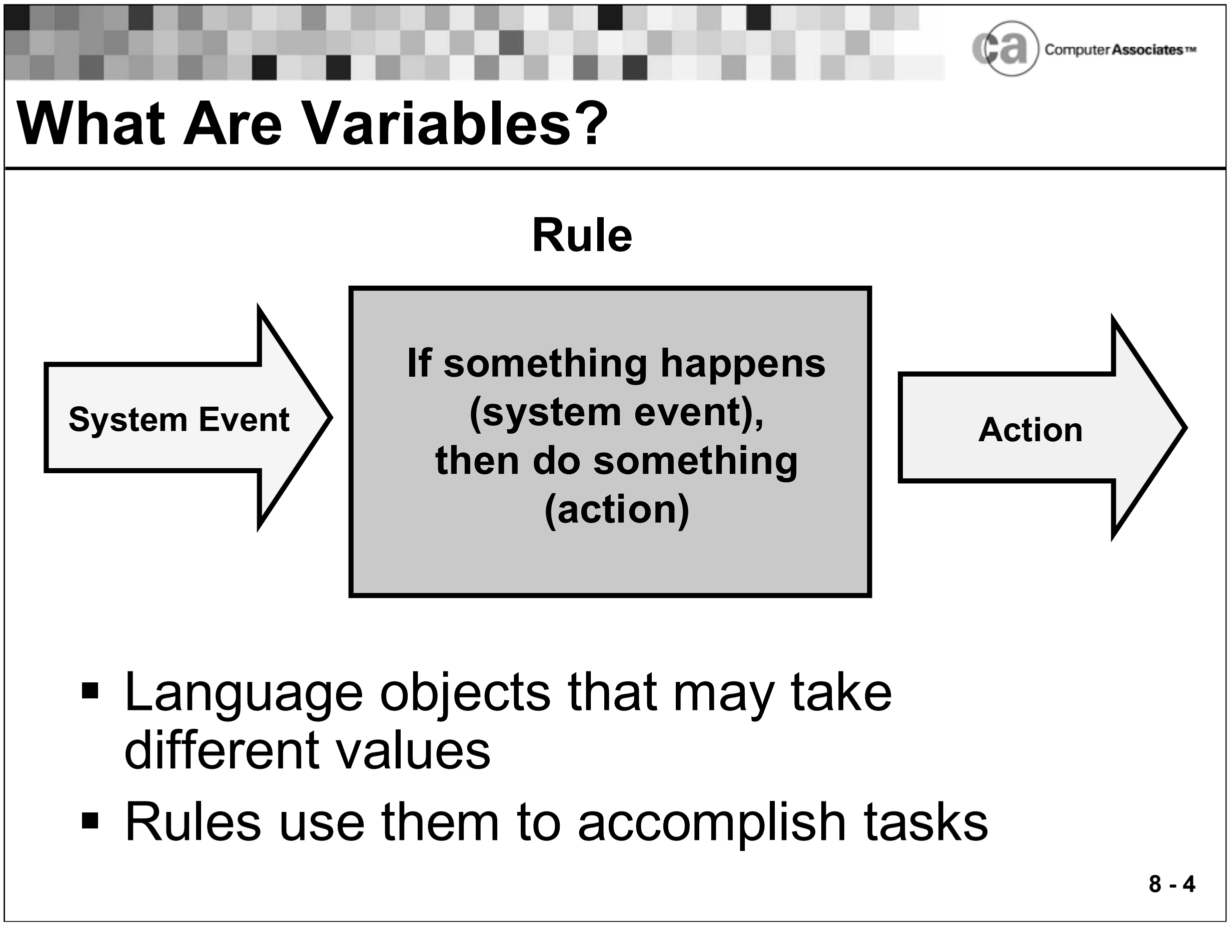
OPSLOG

OPSVIEW

8 - 3

AOF Variables

When you create rules to trigger a particular system event, you may need to know specific information about the event such as the name of the job that issued a message, a message's route codes, or the name of the console that issued a system command. Also, the logic of your particular automated application may require data to be saved over various executions of an event or between different events. You can obtain this type of functionality by using AOF variables.



What Are Variables?

Rule

System Event

If something happens (system event), then do something (action)

Action

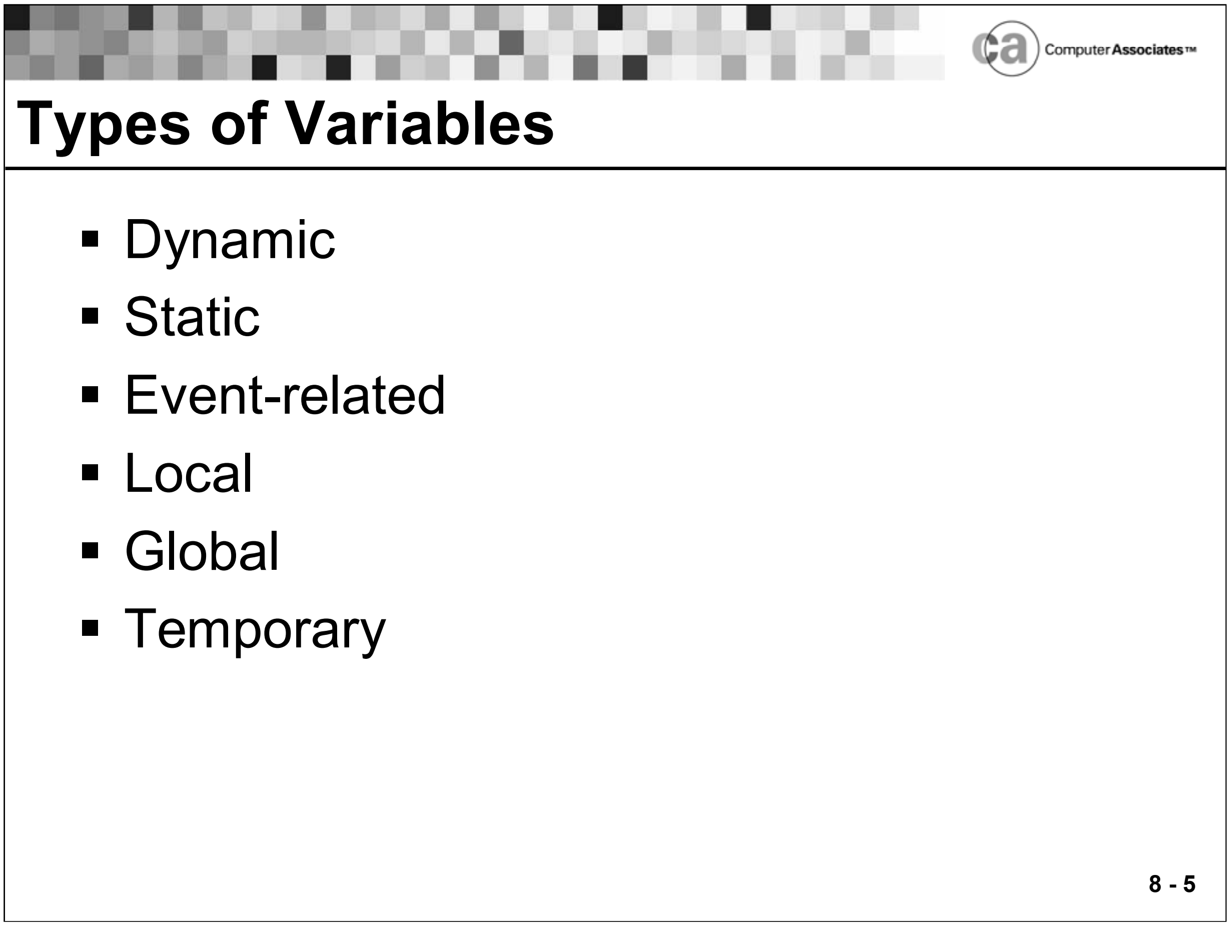
- Language objects that may take different values
- Rules use them to accomplish tasks

8 - 4

What Are Variables?

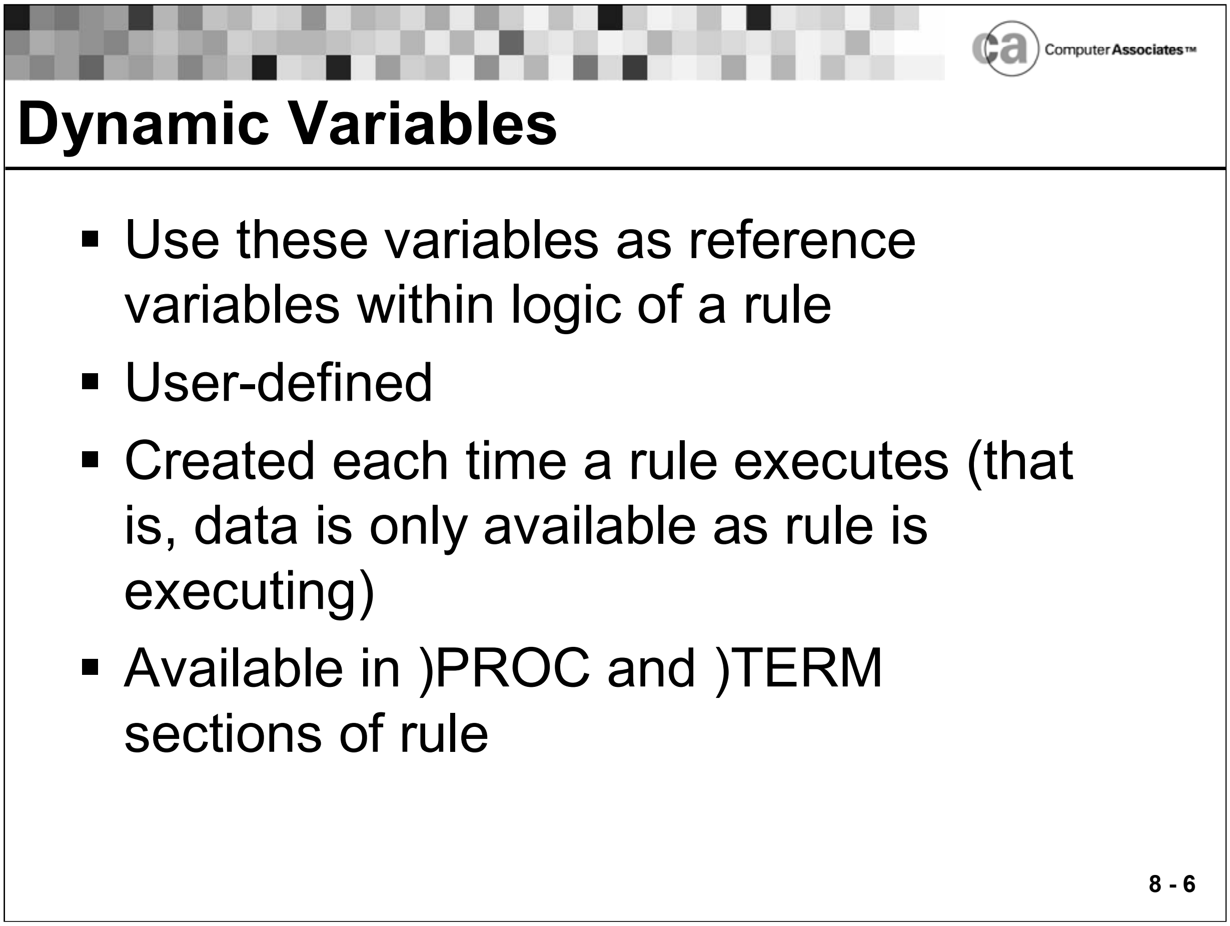
Variables are data items whose values can change while a program is running. To change the value of a variable, you assign a new value to it.

Rules use variables to accomplish tasks, such as retaining data across various executions of a rule.



Types of Variables

- Dynamic
- Static
- Event-related
- Local
- Global
- Temporary



Dynamic Variables

- Use these variables as reference variables within logic of a rule
- User-defined
- Created each time a rule executes (that is, data is only available as rule is executing)
- Available in)PROC and)TERM sections of rule

8 - 6

Dynamic Variables

Dynamic variables are simple variables that are user-defined and created each time a rule executes. The data of a dynamic variable is available only when a rule is executing.

Dynamic variables are:

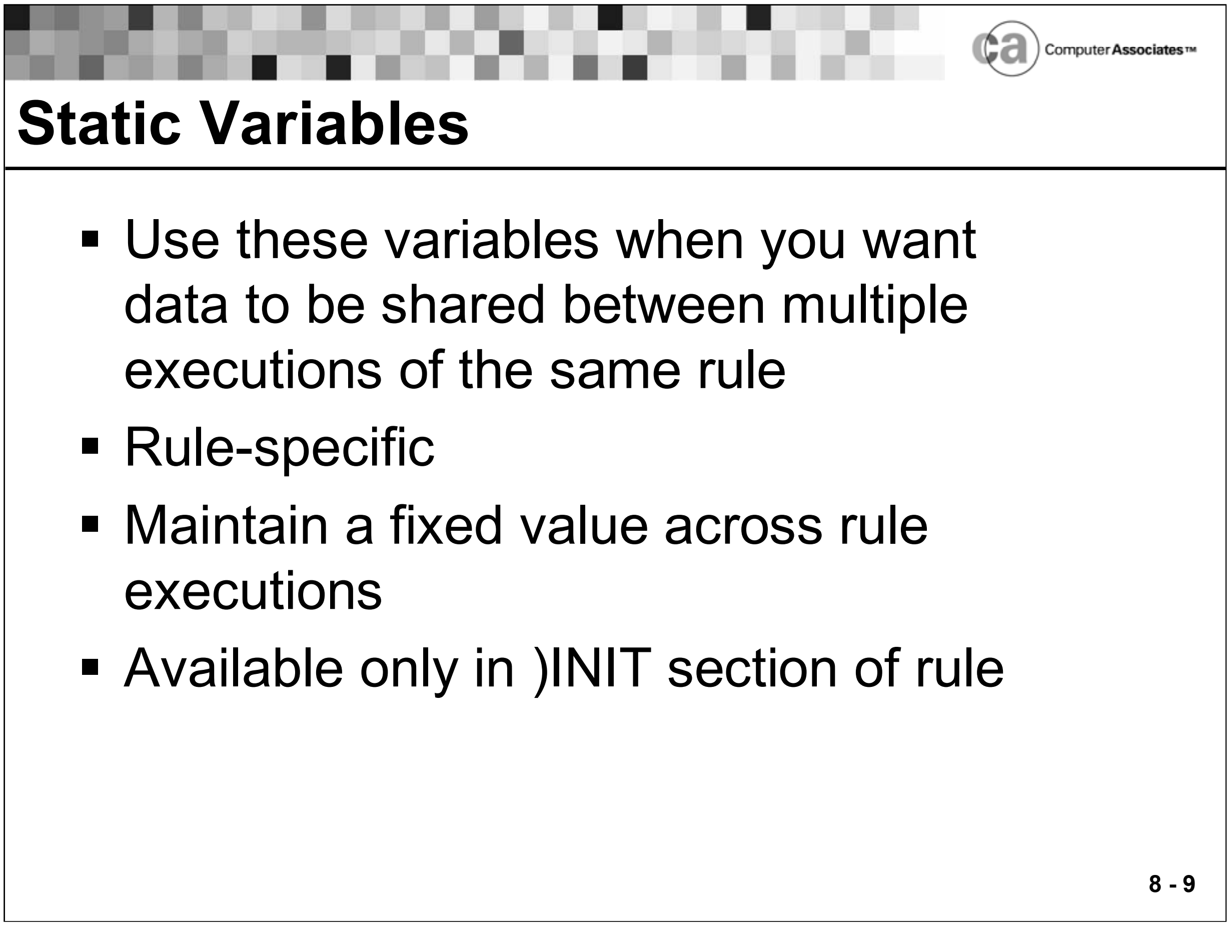
- Simple variables used only in the)PROC or)TERM section of rules.
- Non-global compound symbols.
- Examples of dynamic variable names include COUNT and JOB.COUNT.

Characteristics

- The name can be up to 256 characters in length.
- The value can be up to 32,000 bytes in length.
- They can be compound symbols such as JOB.COUNT (although they should not contain a reserved stem used by event variables such as MSG.xxx or CMD.xxx).

- The value of an uninitialized dynamic variable (a variable that has not yet been assigned a value) is the variable name itself.
- The three special variables in standard REXX (RC, RESULT, and SIGL) are always dynamic variables in OPS/REXX.

8 - 8



Static Variables

- Use these variables when you want data to be shared between multiple executions of the same rule
- Rule-specific
- Maintain a fixed value across rule executions
- Available only in)INIT section of rule

8 - 9

Static Variables

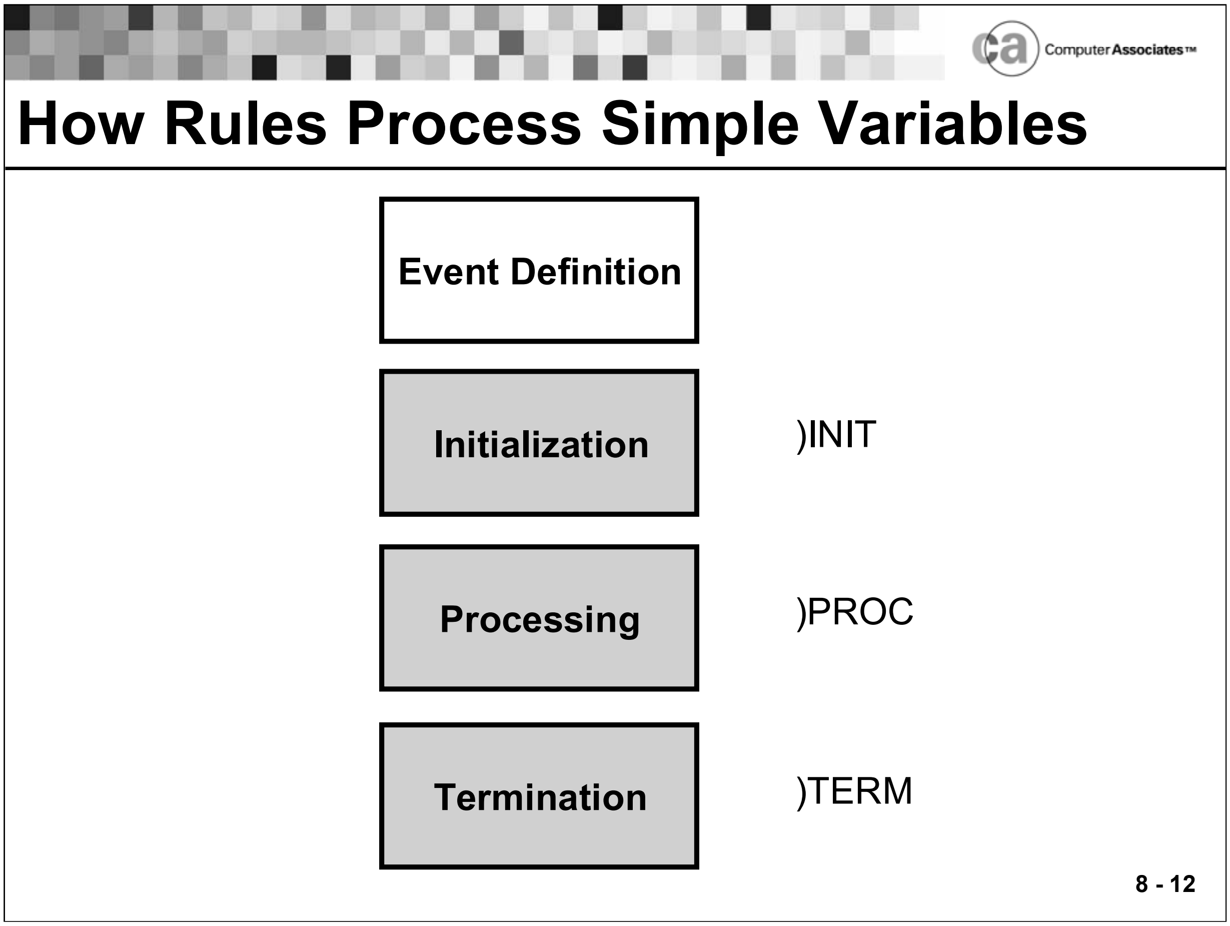
- Static variables maintain a fixed value across multiple executions of a single rule. This means that data can be shared between executions of the same rule.
- Examples of static variable names include XYZ, JOBS, and A5.

Static variables have these characteristics:

- They are only available in the)INIT section of a rule.
- They are rule-specific.
- The same variable name used in the initialization sections of two different rules refers to two different variables.
- They do not allow serial access. Use global variables and the OPSVALUE function if you require serialization.
- The name can be up to 50 characters in length.

- The value can be up to 256 bytes in length. During a rule section's execution, the length of a static variable may grow to the maximum allowed for OPS/REXX variables. However, after the section containing the static variable executes, Unicenter CA-OPS/MVS truncates the static variable's value to 256 bytes.
- They cannot be a compound symbol (for example, JOB.COUNT).
- They are deleted upon rule disablement.
- They cannot be used with the VALUE function.
- They cannot be used in INTERPRET statements.
- They cannot be used as host variables in SQL statements.
- They are ignored when specified on the TRACE VAR instruction.
- Note: If you intend to use status variables to pass data between rule sections or between multiple executions of the rule, the data must be contained within the first 256 bytes of the variable.

8 - 11



How Rules Process Simple Variables

How Rule Sections Process Variables

As you learned earlier, a rule contains sections. Variables are processed in the)INIT,)PROC, and)TERM sections of a rule.

Variable Processing in the)INIT Section

- When the)INIT section of any rule references a simple variable (a variable that is not a stem variable):

  The variable keeps its value between executions of a rule.

  If a variable name referenced in the)INIT section also appears in the)PROC and/or)TERM section of a rule, Unicenter CA-OPS/MVS associates all references with the same variable.

Variable Processing in the)PROC Section

- When the)PROC section of any rule references a simple variable:

  If the)INIT section of the rule also referenced this variable, Unicenter CA-OPS/MVS treats the variable as static, and the variable keeps the value it received in the)INIT section or during a previous execution of the rule.

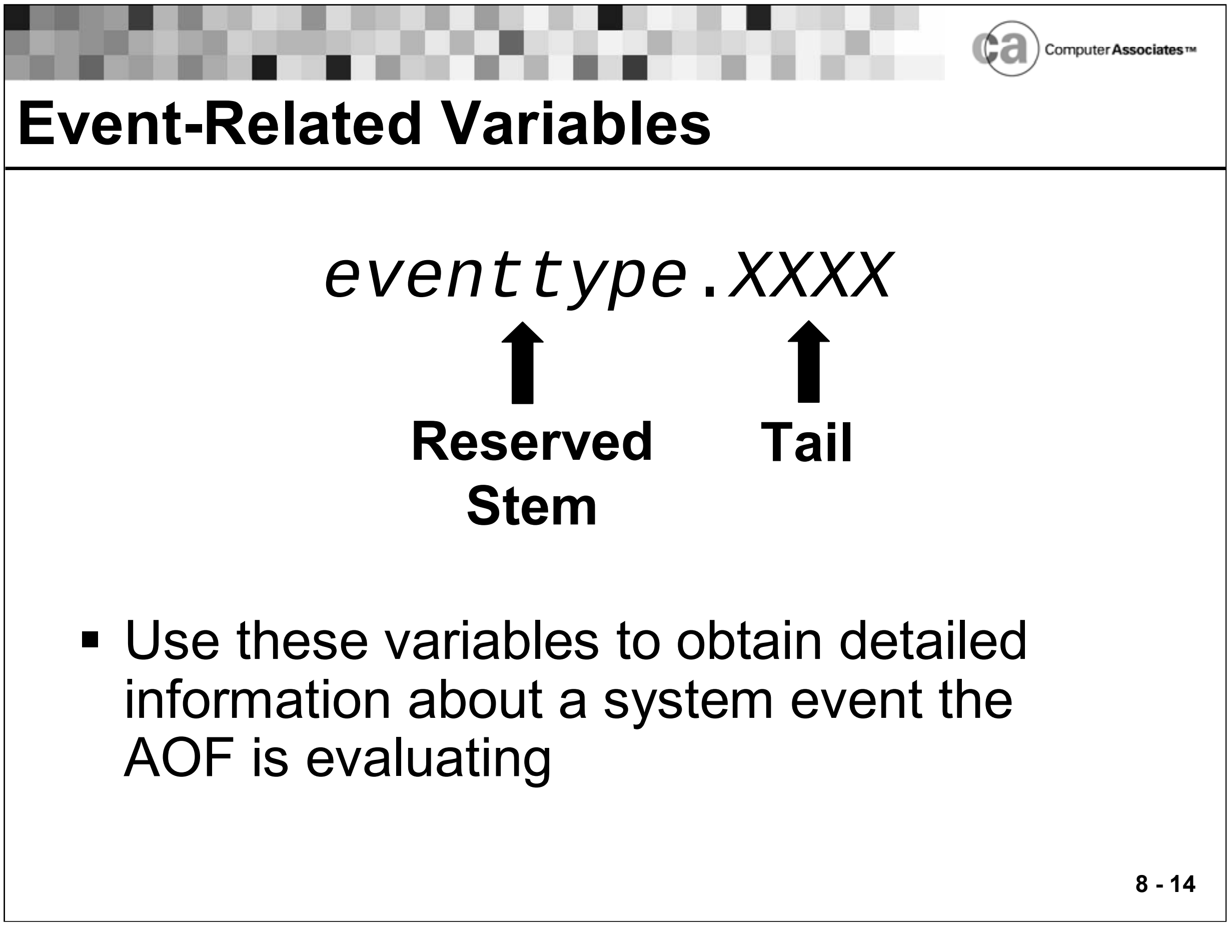
  If the)INIT section of the rule does not reference the same variable, Unicenter CA-OPS/MVS considers the variable dynamic and reinitializes it each time the)PROC section executes.

Variable Processing in the)TERM Section

- When the)TERM section of any rule references a simple variable:

  If the)INIT section of the rule also referenced this variable, Unicenter CA-OPS/MVS treats the variable as static, and the variable keeps the value it received in the)INIT section or during a previous execution of the rule.

  If the)INIT section of the rule does not reference the same variable, Unicenter CA-OPS/MVS considers the variable dynamic and sets the variable to the value of its name.



Event-Related Variables

eventtype.XXXX

Reserved Stem

Tail

- Use these variables to obtain detailed information about a system event the AOF is evaluating

8 - 14

Event-Related Variables

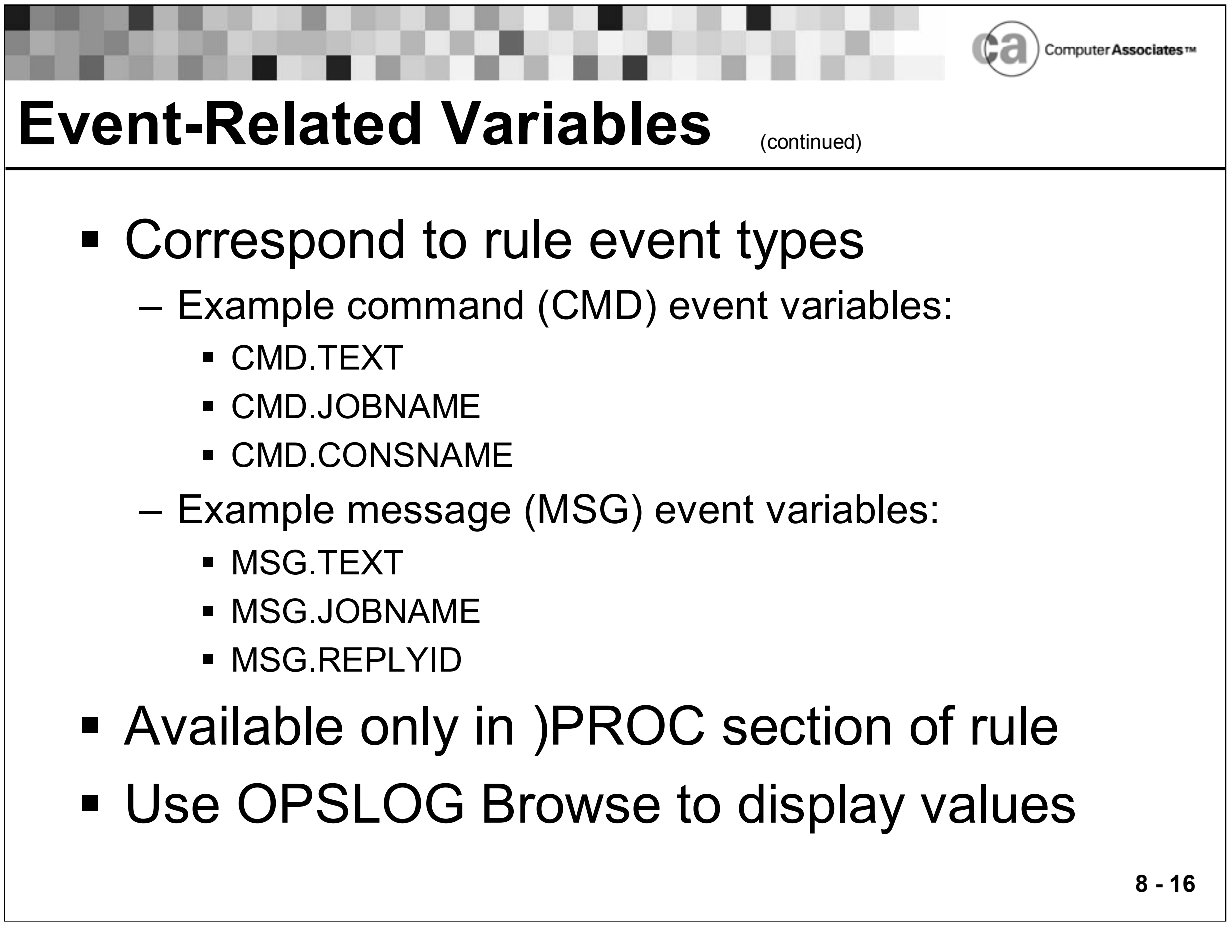
Event-related variables contain information about specific system events. For example, a variable called MSG.TEXT can store the text of a specific WTO event.

Event-related variables correspond to the rule event types:

- Automatic Restart Management variables (ARM)
- Application Program Interface (API)
- Command event variables (CMD)
- Delete-operator-message event variables (DOM)
- End-of-job event variables (EOJ)
- End-of-memory event variables (EOM)
- End-of-step event variables (EOS)
- Variable events for global variables (GLV)
- Message event variables (MSG)
- OMEGAMON event variables (OMG)
- Request event variables (REQ)

- Security event variables (SEC)
- Screen event variables (SCR)
- Time limit excession event variables (TLM)
- Time-of-day event variables (TOD)
- UNIX System Services event variables (USS)

For detailed information about each event-related variable according to event type, see your Unicenter CA-OPS/MVS documentation.



Event-Related Variables (continued)

- Correspond to rule event types
  - Example command (CMD) event variables:
    - CMD.TEXT
    - CMD.JOBNAME
    - CMD.CONSNAME
  - Example message (MSG) event variables:
    - MSG.TEXT
    - MSG.JOBNAME
    - MSG.REPLYID
- Available only in)PROC section of rule
- Use OPSLOG Browse to display values

8 - 16

Event-Related Variables

As you just learned, event-related variables correspond to rule event types. This section discusses some of the more commonly used variables. For detailed information about each type of variable, see your documentation.

Command Event Variables

- Within the command (CMD) rule type, you will have access to various CMD.xxxx variables. Some popular command event variables are:

  CMD.TEXT

  Contains the text of the current command.

  CMD.JOBNAME

  Contains the name of the job that issued the command.

  CMD.CONSNAME

  Contains the name of the console that issued the command.

Message Event Variables

- Within the message (MSG) rule type, you will have access to various MSG.xxxx variables. Some popular message event variables are:

  MSG.TEXT

  Contains the text of the current message.

  MSG.JOBNAME

  Contains the name of the job that issued the message.

  MSG.REPLYID

  Contains the reply ID of WTOR messages.

Event-related variables have these characteristics:

- They are available only in the)PROC section of a rule.
- They are automatically provided by the AOF engine.
- Changes made to event-related variables change the corresponding information about the event itself.

  Example: Changing the contents of the MSG.TEXT variable in a message rule changes the text of the WTO message. Modifying the CMD.TEXT variable in a command rule changes the original command that was entered.

  Note: Some event-related variables are designated as read-only; changing the value of a read-only variable does not cause an error, but the change has no effect. Because the values of read-only event-related variables do not change, all rules that execute for a single event have the same event data.
- Changes to an event-related variable by multiple rules are cumulative. The first rule to execute receives original event information. Subsequent rules (executing in response to the same event) receive event information modified by each preceding rule.

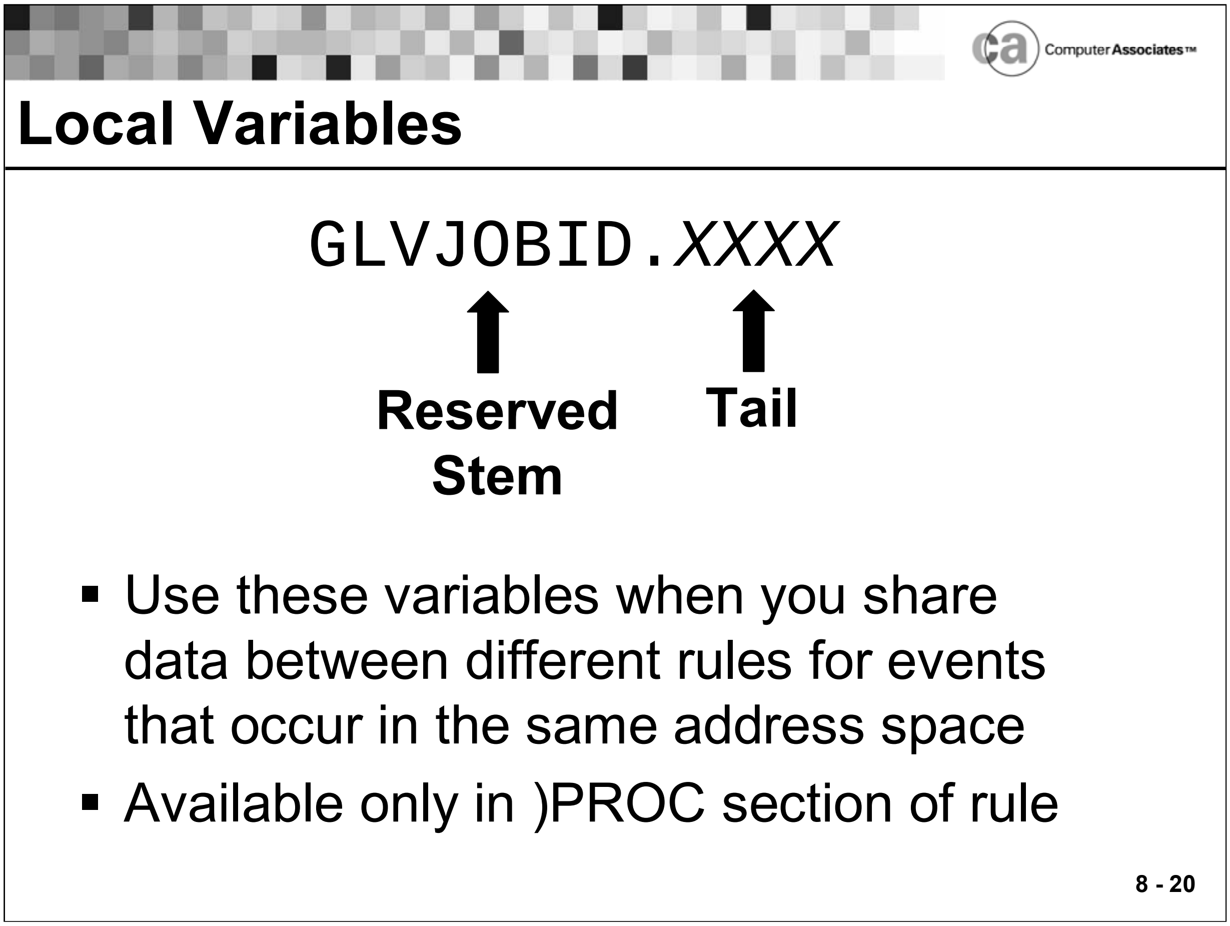
  Note: Because rules cannot change the value of a read-only variable, the variable always contains original event information.
- All rules that would normally execute in response to an event do so regardless of how each rule changes an event-related variable.

  Example: If a message rule changes the message ID contained in the MSG.ID variable, all rules matching the original message ID still execute (and no rules matching the new message ID execute).
- You can display the values of most event-related variables in OPSLOG Browse via the DISPLAY command.

Changes to Event-related Variables

- Changing the value of some event-related variables also changes the event. For example, changing the value of the MSG.TEXT variable alters the text of the associated WTO. Other “read only” event-related variables cannot be modified.
- Changes to modifiable event-related variables are cumulative. The first rule that an event triggers gets the original event information; rules for the same event executing later get modified copies of this information. Even if a rule modifies an event-related variable, all rules eligible to execute for an event still execute. For instance, if a message rule modifies the variable MSG.TEXT, all rules referencing the original message text execute, but rules referencing the revised text do not execute.
- Because the values of read-only event-related variables do not change, all rules that execute for a single event get the same event data.

8 - 19



Local Variables

GLVJOBID.*XXXX*

Reserved Stem

Tail

- Use these variables when you share data between different rules for events that occur in the same address space
- Available only in)PROC section of rule

8 - 20

Local Variables

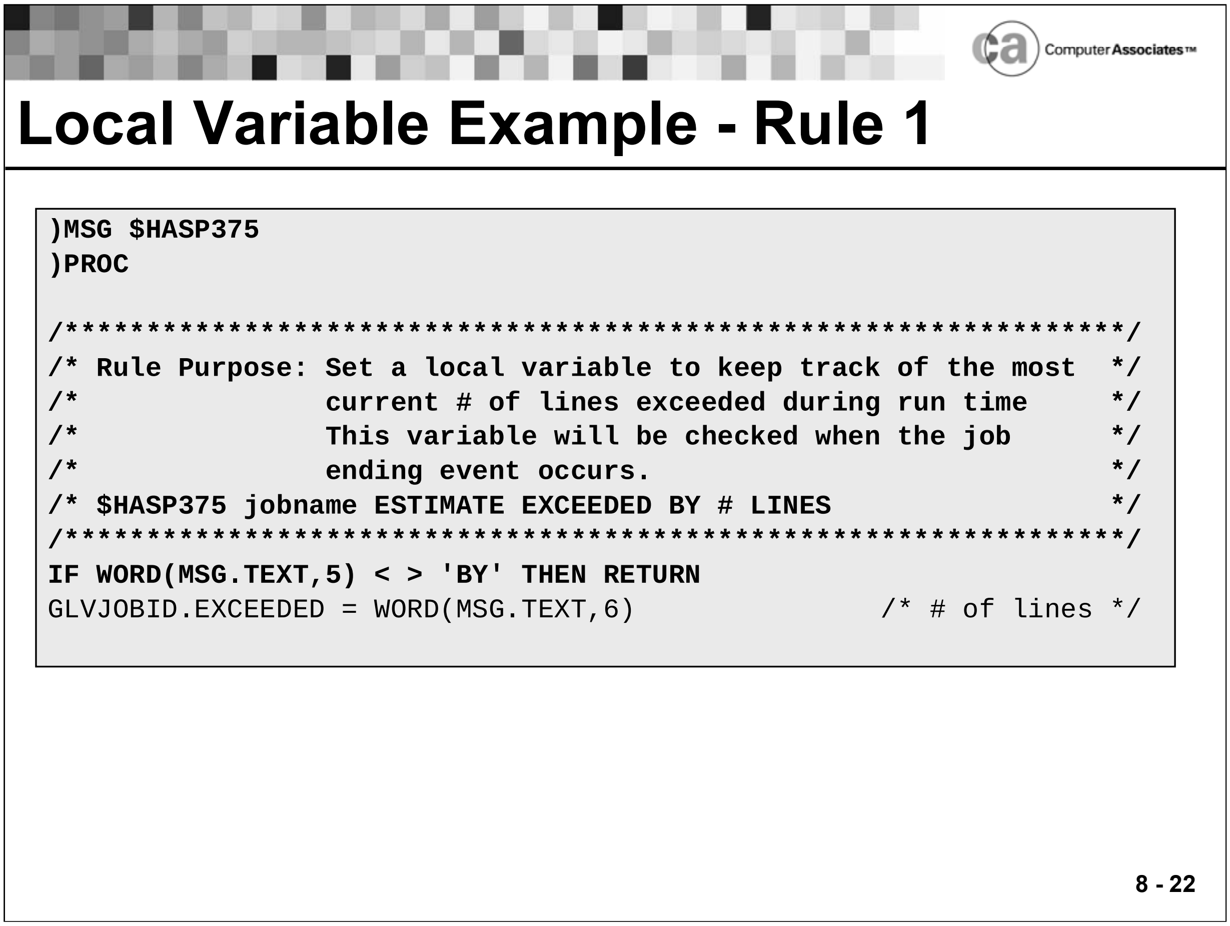
Local variables are compound symbols that begin with a reserved stem of GLVJOBID. They allow you to share data between different rules for events that occur in the same address space. This enables you to save the data generated during one event created by a job and then use that data within another event created by the same job.

The stem is GLVJOBID, up to and including the first period. The stem is followed by the tail.

Characteristics

- They are only available in the)PROC section of a rule.
- They are unique to the address space that triggered the rule.
- The name can be up to 78 characters in length, including the GLVJOBID stem.
- The value can be up to 32,000 bytes in length.

- They are automatically deleted by Unicenter CA-OPS/MVS when the address space with which they are associated terminates.
- Some messages appear to be issued from a particular job, but they are actually issued on behalf of another address space. For example, the $HASP100 message that indicates a job is on the internal reader is actually issued from the JES address space, not the job's address space. Before selecting a GLVJOBID variable, use the OPSLOG Browse JOBNAME column to verify that the message is being issued from a unique address space.



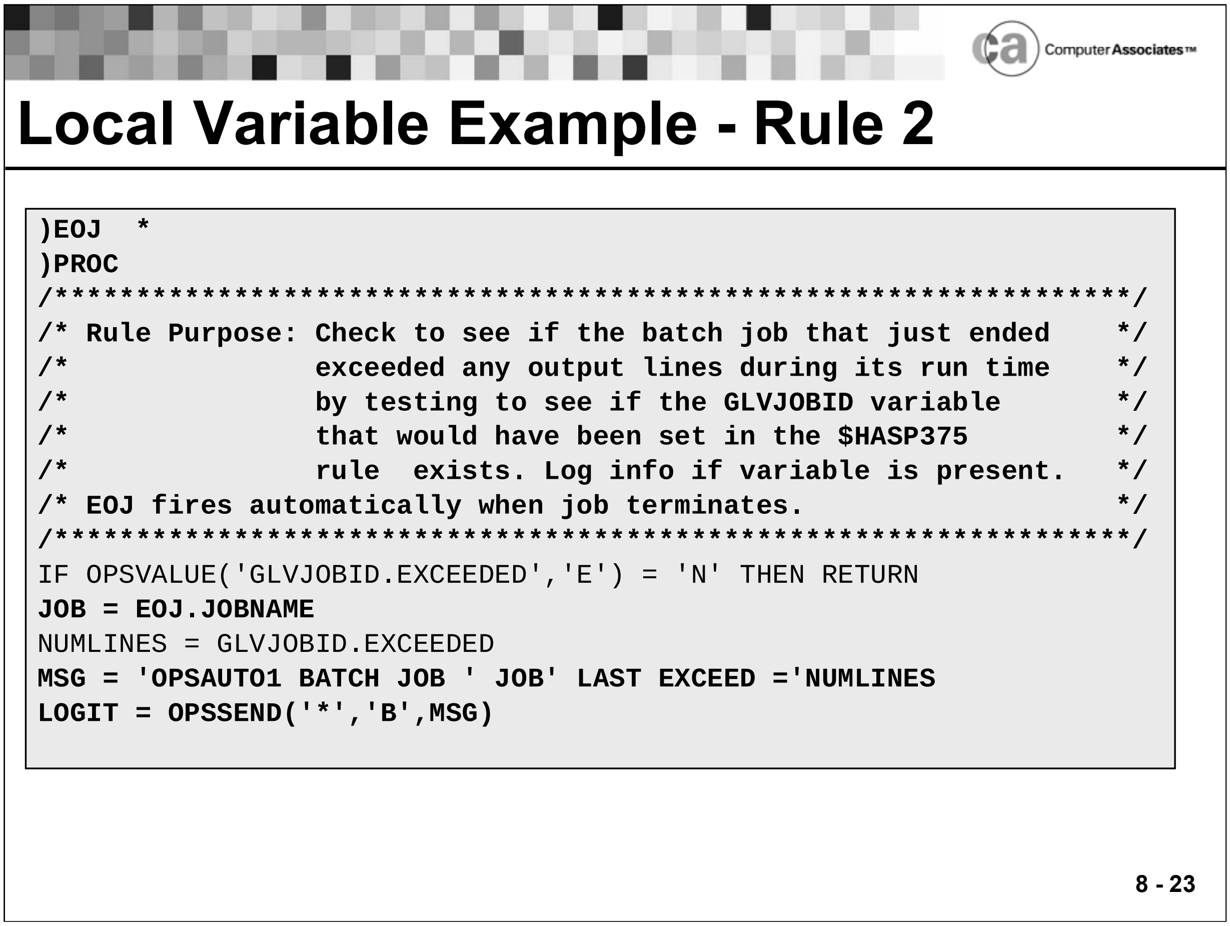
Local Variable Example - Rule 1

```
)MSG $HASP375
)PROC

/*****************************************************************/
/* Rule Purpose: Set a local variable to keep track of the most  */
/*               current # of lines exceeded during run time     */
/*               This variable will be checked when the job      */
/*               ending event occurs.                            */
/* $HASP375 jobname ESTIMATE EXCEEDED BY # LINES                 */
/*****************************************************************/
IF WORD(MSG.TEXT,5) < > 'BY' THEN RETURN
GLVJOBID.EXCEEDED = WORD(MSG.TEXT,6)               /* # of lines */
```

8 - 22

Local Variable Example - Rule 1

The above rule logic sets a local variable, GLVJOBID.EXCEEDED, that keeps track of the number of output lines that were exceeded during the job’s run time; the variable is checked when the event that ends the job occurs.



Local Variable Example - Rule 2

```
)EOJ  *
)PROC
/*****************************************************************/
/* Rule Purpose: Check to see if the batch job that just ended    */
/*               exceeded any output lines during its run time    */
/*               by testing to see if the GLVJOBID variable       */
/*               that would have been set in the $HASP375         */
/*               rule  exists. Log info if variable is present.   */
/* EOJ fires automatically when job terminates.                   */
/*****************************************************************/
IF OPSVALUE('GLVJOBID.EXCEEDED','E') = 'N' THEN RETURN
JOB = EOJ.JOBNAME
NUMLINES = GLVJOBID.EXCEEDED
MSG = 'OPSAUTO1 BATCH JOB ' JOB' LAST EXCEED ='NUMLINES
LOGIT = OPSSEND('*','B',MSG)
```

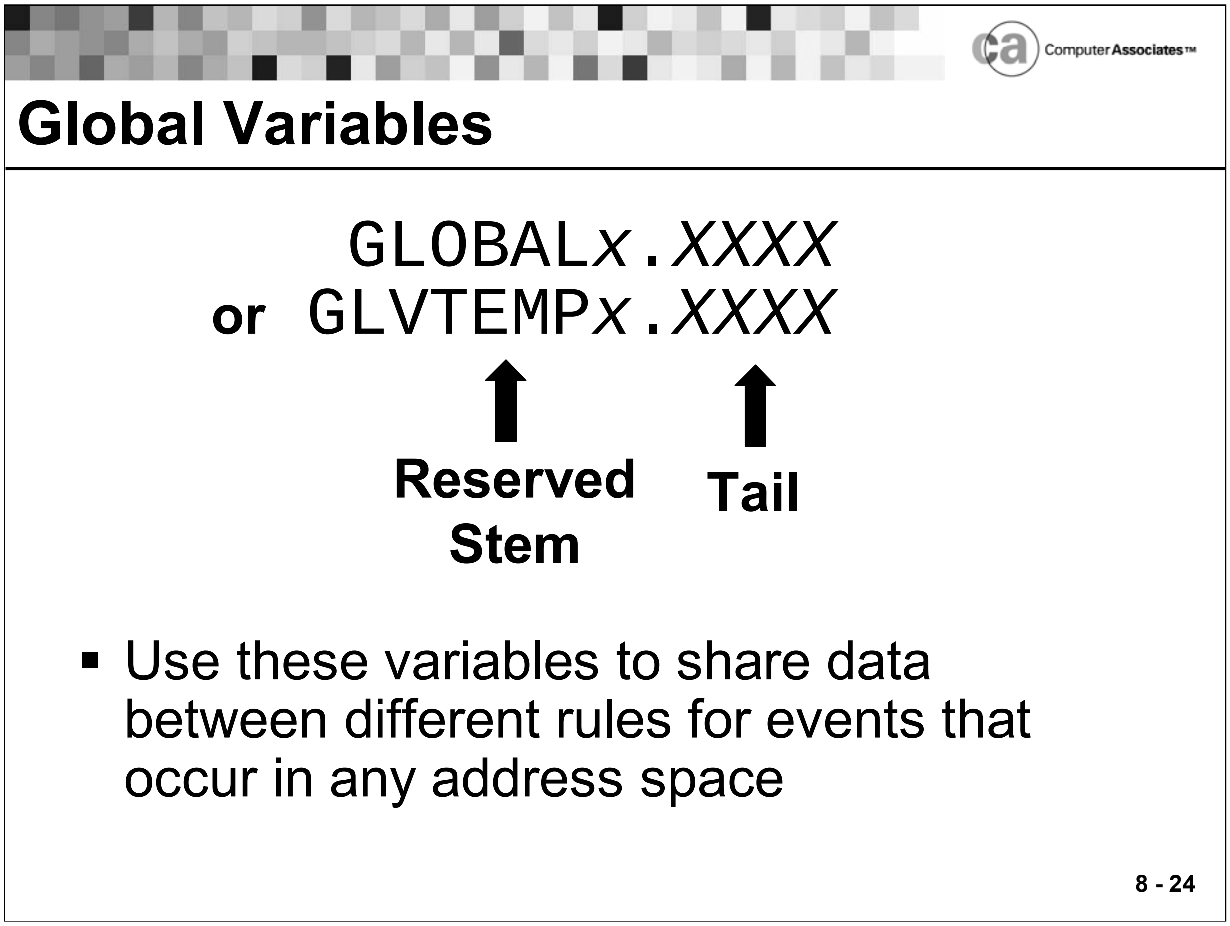
Local Variable Example - Rule 2

The above rule logic determines whether the batch job that just ended exceeded the expected number of output lines. It tests whether the GLVJOBID variable that would have been set in the $HASP375 rule exists.

Suppose JOB1 and JOB2 start on the system at the same time. JOB1 exceeds the number of expected output lines, causing the JOB1 address space to produce a $HASP375 message. This executes Rule 1, which sets a unique GLVJOBID.EXCEEDED variable for JOB1 only.

Both JOB1 and JOB2 end at the same time, executing Rule 2. While Rule 2 is processing the EOJ event that was triggered by the end of JOB1, the local variable GLVJOBID will exist and the rule will generate an informational message in the OPSLOG. This message indicates the value of the GLVJOBID.EXCEEDED local variable that was set in Rule 1.

The check against the GLVJOBID.EXCEEDED variable in Rule 2 while processing the EOJ event that was triggered by the end of JOB2 will not exist, causing the logic to simply exit the rule.



Global Variables

GLOBAL*x*.*XXXX*
or GLVTEMP*x*.*XXXX*

Reserved Stem

Tail

- Use these variables to share data between different rules for events that occur in any address space

8 - 24

Global Variables

- Global variables allow data to be shared between different rules for events that occur in any address space. Global variables begin with a stem of GLOBALx or GLVTEMPx. A stem of GLOBALx causes the variable to be permanently saved across IPLs or cycles of Unicenter CA-OPS/MVS. A stem of GLVTEMPx causes the variable to be saved only while Unicenter CA-OPS/MVS is active.
- OPS/REXX global variables allow communication between two or more rules or between rules and TSO applications.
- OPS/REXX global variables used in AOF rules can be accessed from programs running in OPSOSF servers, batch, TSO, NetView, and UNIX System Services environments.
- Examples of global variable names include GLOBAL.A.B and GLOBAL.MAIN.
- The stem is GLOBALx or GLVTEMPx, up to and including the first period. The stem is followed by the tail.

Global variables have these characteristics:

- They are OPS/REXX compound symbols containing any of these stems:

  GLOBAL.

  GLOBALx.

  GLVTEMPx.

- They are not declared. Any use of a variable name containing one of the valid GLOBAL stems creates a global variable.
- They can be modified using ordinary assignment statements.
- They trigger global events if their values change.

  Note: Changing global variables with stems of GLOBAL0 through GLOBAL9 does not trigger global variable events.

- They have derived names that can be up to 84 bytes in length.

  Example: Suppose that an OPS/REXX program has initialized the simple symbol "I" to 3, and that you have coded this assignment statement:

  ```
  GLOBAL.I = 'THIS IS A TEST VALUE'
  ```

- The global variable that the assignment statement creates is GLOBAL.3.
- The value can be up to 32,000 bytes in length.

  Note: Global variables containing up to 76 bytes use much less storage space than those containing more.

- They are case-sensitive (after the stem) when used in an OPSVALUE function.

  Example: The OPSVALUE function recognizes GLOBAL.ABC and GLOBAL.abc as two different variables.

- They are visible from 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 or TSO-based OPS/REXX program.
- They do not allow serial access. Use the OPSVALUE function if you require serialization.
- You create global variables within an OPS/REXX program by naming them in an assignment statement.

  Note: In a GLOBALx. or GLVTEMPx. stem, the value of x can be a single letter (A through Z) or number (0 through 9). If the value of x is a letter, changing the value of the global variable triggers a global variable rule. If you do not intend to trigger a rule, use a numeric value for x; processing will be faster.

- There are two types of global variables: standard and temporary. Standard global variables are nonvolatile; they are checkpointed to data-in-virtual data sets. Temporary global variables are not checkpointed and they do not exist across IPLs or restarts of Unicenter CA-OPS/MVS. The stem GLVTEMPx identifies temporary global variables.
- A global variable name is case-sensitive when used in the OPSVALUE function. For example, Unicenter CA-OPS/MVS treats variables named GLOBAL.ABC and GLOBAL.abc as two separate variables.
- Until you initialize a global variable, its value and name are the same. For instance, an uninitialized variable named GLOBAL.DOG has the value "GLOBAL.DOG."
- However, be careful when initializing the stem GLOBAL., because doing so:

  Deletes any global variables that existed before you initialized the stem.

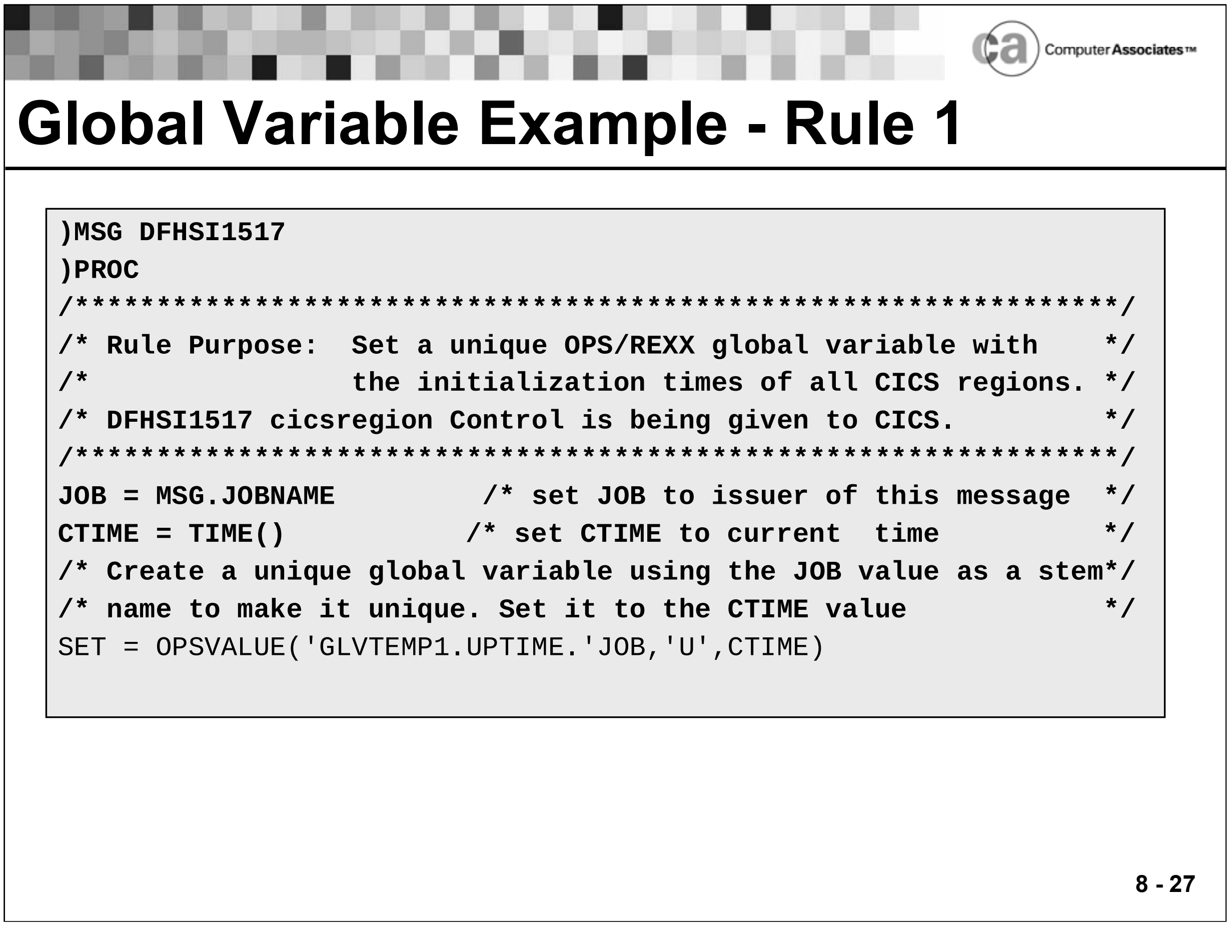
  Assigns the initial stem value to any new global variables created after the stem was initialized.

Changes to Global Variables

- Once you create a global variable, you can modify it using ordinary assignment statements. To delete the variable, you use the OPSVALUE built-in function of OPS/REXX.
- Changing the value of a global variable triggers a global variable event unless that variable has a stem of GLOBALx. or GLVTEMPx. (where x is a number from 0 to 9).

Accessing Global Variables

- All OPS/REXX programs, including rules and TSO-based programs, can access global variables. Therefore, you can set a global variable value in a rule and then check or reset it in another rule or an OPS/REXX program.
- Access to a global variable is not serialized when an OPS/REXX instruction references it. Use the OPSVALUE function when serialization is required.
- If a TSO-based OPS/REXX program references a global variable while Unicenter CA-OPS/MVS is down, a run-time error occurs. (AOF rules do not run when Unicenter CA-OPS/MVS is inactive.)
- To view and/or modify the current values of global variables, use OPSVIEW option 4.8.



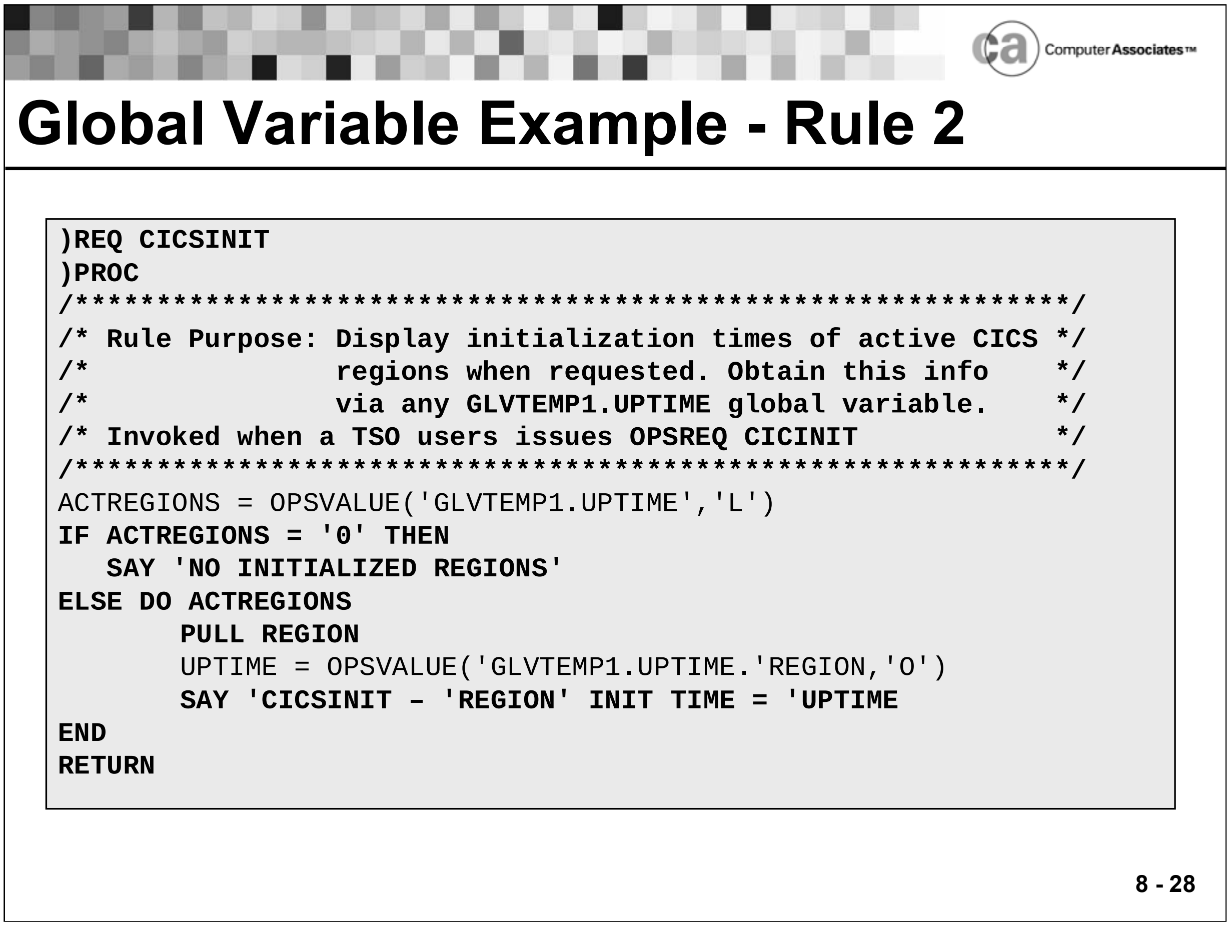
Global Variable Example - Rule 1

```
)MSG DFHSI1517
)PROC
/****************************************************************/
/* Rule Purpose:  Set a unique OPS/REXX global variable with    */
/*                the initialization times of all CICS regions. */
/* DFHSI1517 cicsregion Control is being given to CICS.         */
/****************************************************************/
JOB = MSG.JOBNAME         /* set JOB to issuer of this message  */
CTIME = TIME()           /* set CTIME to current  time          */
/* Create a unique global variable using the JOB value as a stem*/
/* name to make it unique. Set it to the CTIME value            */
SET = OPSVALUE('GLVTEMP1.UPTIME.'JOB,'U',CTIME)
```

8 - 27

Global Variable Example

The above rule executes for every DFHSI1517 message that is issued when a CICS region initiates. An OPS/REXX global variable is created, using the region name as part of the stem name so that a unique variable exists for each CICS region (for example, GLVTEMP1.UPTIME.CICSA for region CICSA and GLVTEMP1.UPTIME.CICSB for region CICSB). The variable is set to the current system time.



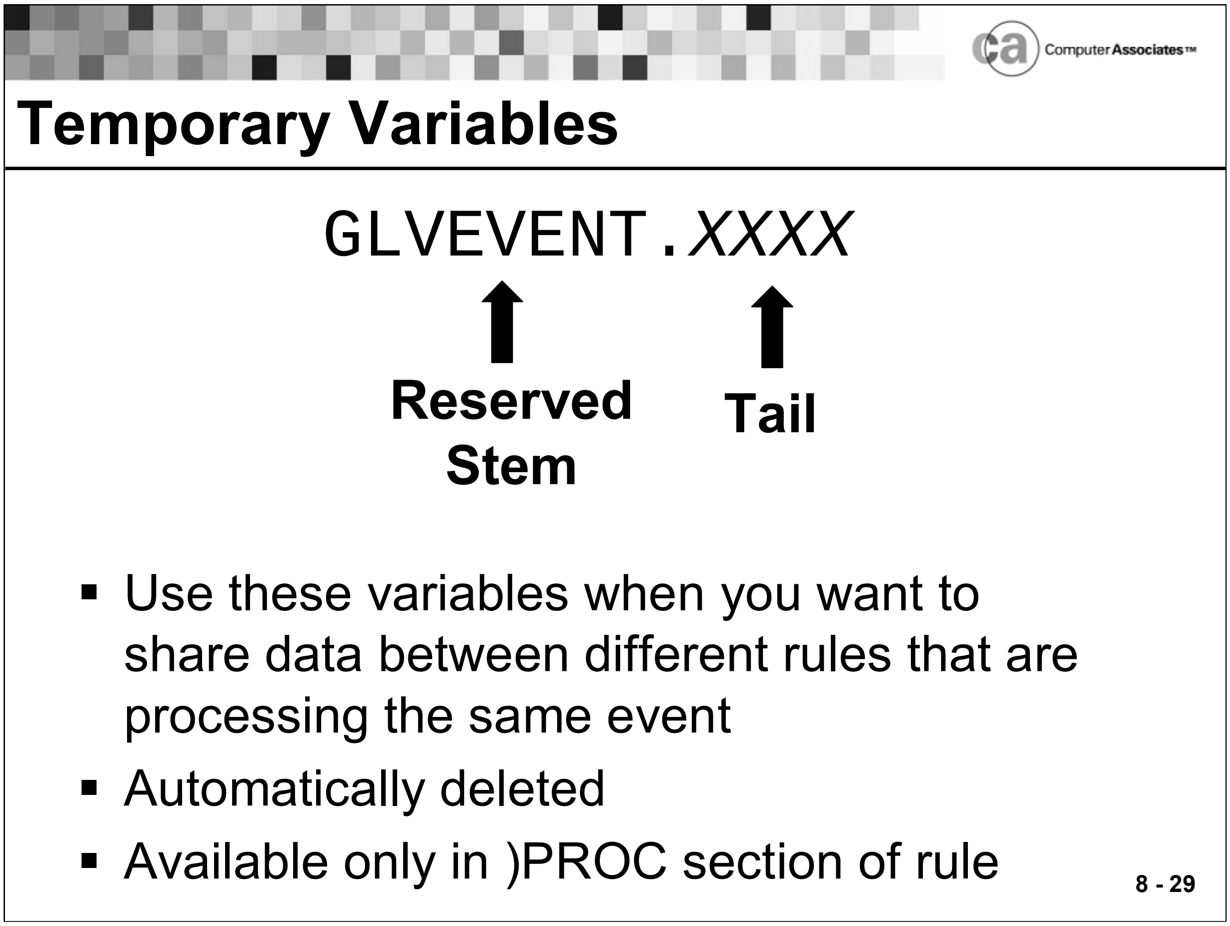
Global Variable Example - Rule 2

```
)REQ CICSINIT
)PROC
/*************************************************************/
/* Rule Purpose: Display initialization times of active CICS */
/*               regions when requested. Obtain this info    */
/*               via any GLVTEMP1.UPTIME global variable.    */
/* Invoked when a TSO users issues OPSREQ CICINIT            */
/*************************************************************/
ACTREGIONS = OPSVALUE('GLVTEMP1.UPTIME','L')
IF ACTREGIONS = '0' THEN
   SAY 'NO INITIALIZED REGIONS'
ELSE DO ACTREGIONS
       PULL REGION
       UPTIME = OPSVALUE('GLVTEMP1.UPTIME.'REGION,'O')
       SAY 'CICSINIT – 'REGION' INIT TIME = 'UPTIME
END
RETURN
```

8 - 28

Global Variable Example

The above rule is a request rule that executes on demand by any TSO user. It has a different event from Rule 1. This rule can access and display the GLVTEMP1.UPTIME OPS/REXX variables set by Rule 1. It uses the OPSVALUE OPS/REXX function to interrogate variable information.



Temporary Variables

GLVEVENT.*XXXX*

Reserved Stem
Tail

- Use these variables when you want to share data between different rules that are processing the same event
- Automatically deleted
- Available only in)PROC section of rule

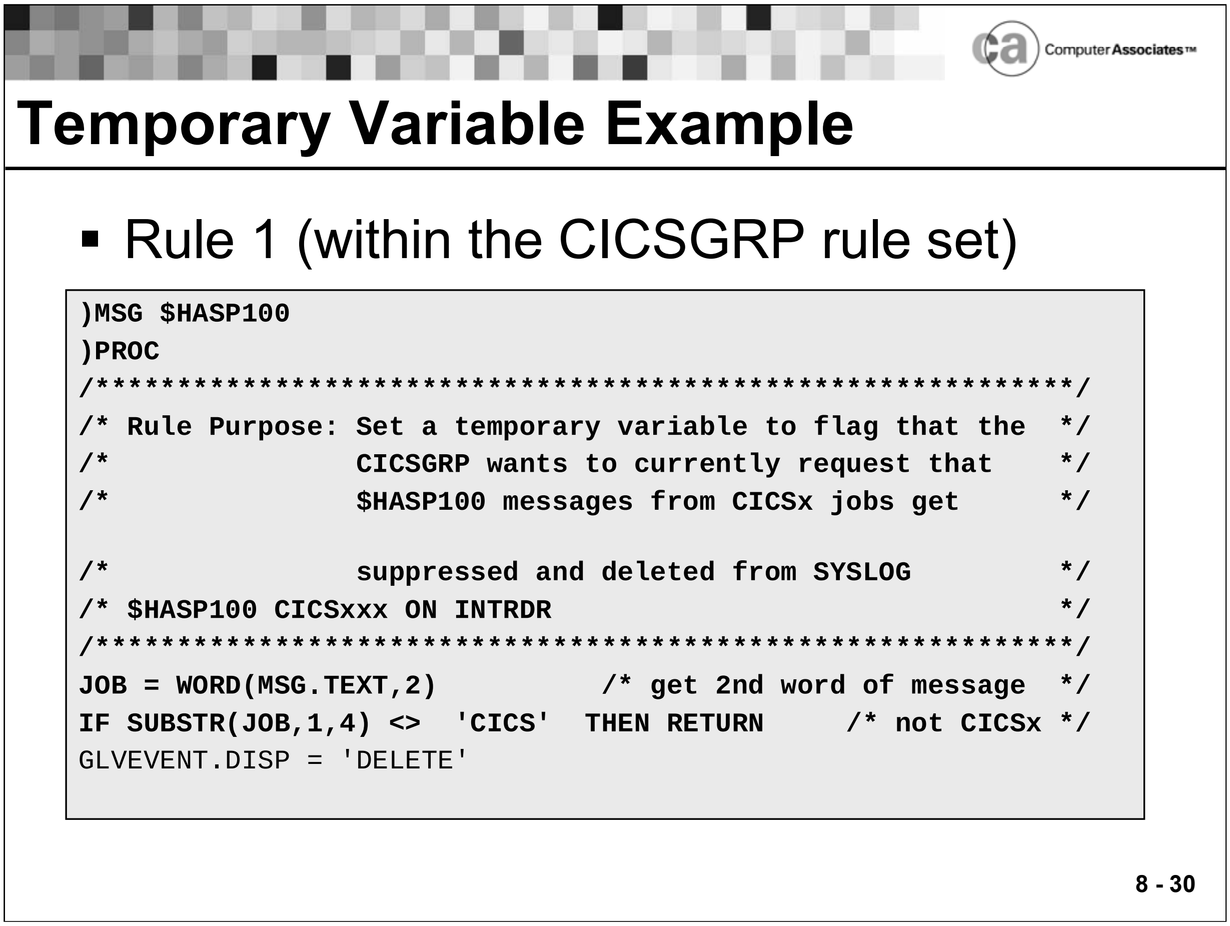
8 - 29

Temporary Variables

- Temporary variables allow data to be shared between different rules that are processing the same event. They begin with the stem GLVEVENT. If you allow different groups (for example, operations, CICS, IMS) to have their own rule sets, you may need to coordinate some process between two or more rules that execute on the same event.
- The stem is GLVEVENT, up to and including the first period. The stem is followed by the tail.

Temporary variables have these characteristics:

- They are available only in the)PROC section of a rule.
- They are unique to rules that execute on the same event.
- The name can be up to 78 characters in length, including the GLVEVENT stem.
- The value can be up to 32,000 bytes in length.
- They are automatically deleted by Unicenter CA-OPS/MVS when the last rule that executes on the event completes.



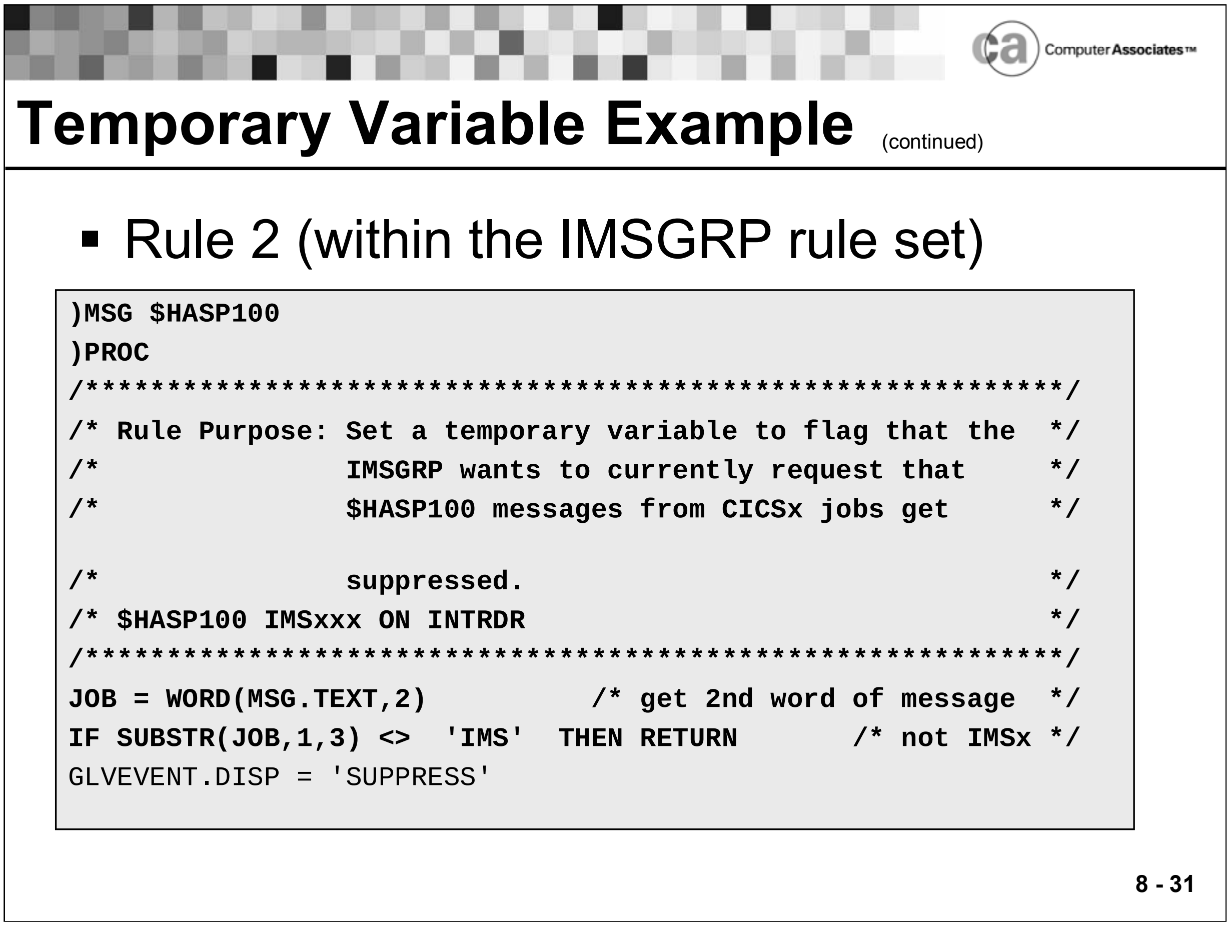
Temporary Variable Example

- Rule 1 (within the CICSGRP rule set)

```
)MSG $HASP100
)PROC
/************************************************************/
/* Rule Purpose: Set a temporary variable to flag that the  */
/*               CICSGRP wants to currently request that    */
/*               $HASP100 messages from CICSx jobs get      */

/*               suppressed and deleted from SYSLOG         */
/* $HASP100 CICSxxx ON INTRDR                                */
/************************************************************/
JOB = WORD(MSG.TEXT,2)          /* get 2nd word of message  */
IF SUBSTR(JOB,1,4) <>  'CICS'  THEN RETURN     /* not CICSx */
GLVEVENT.DISP = 'DELETE'
```

Temporary Variable Example

The above rule sets a temporary variable flag that indicates that CICSGRP has requested that all $HASP100 messages from CICSx jobs be suppressed and deleted from the SYSLOG.



Temporary Variable Example (continued)

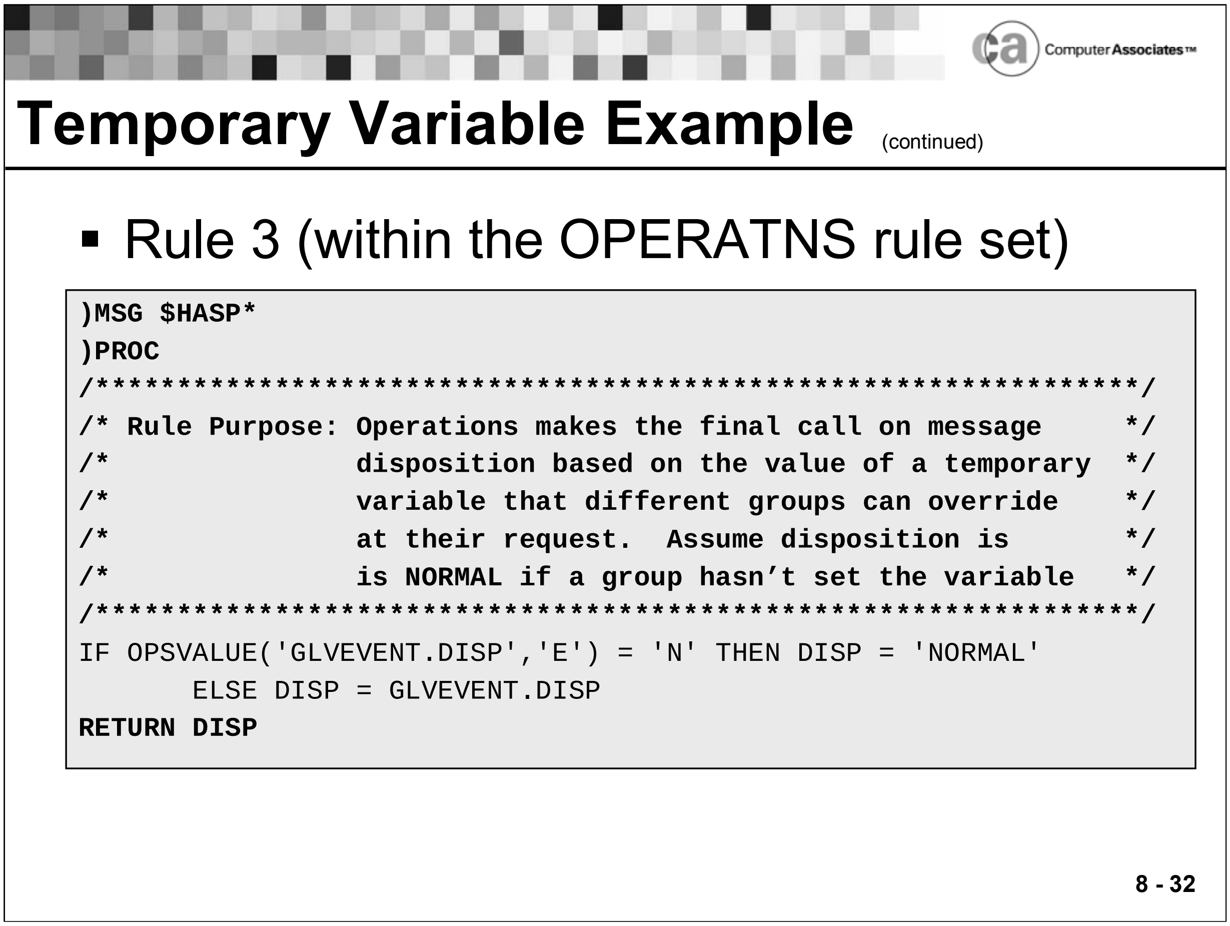
- Rule 2 (within the IMSGRP rule set)

```
)MSG $HASP100
)PROC
/************************************************************/
/* Rule Purpose: Set a temporary variable to flag that the  */
/*               IMSGRP wants to currently request that     */
/*               $HASP100 messages from CICSx jobs get      */

/*               suppressed.                                */
/* $HASP100 IMSxxx ON INTRDR                                */
/************************************************************/
JOB = WORD(MSG.TEXT,2)          /* get 2nd word of message  */
IF SUBSTR(JOB,1,3) <>  'IMS'  THEN RETURN       /* not IMSx */
GLVEVENT.DISP = 'SUPPRESS'
```

8 - 31

Temporary Variable Example

The above rule sets a temporary variable flag that indicates that IMSGRP has requested that all $HASP100 messages from CICSx jobs be suppressed.



Temporary Variable Example (continued)

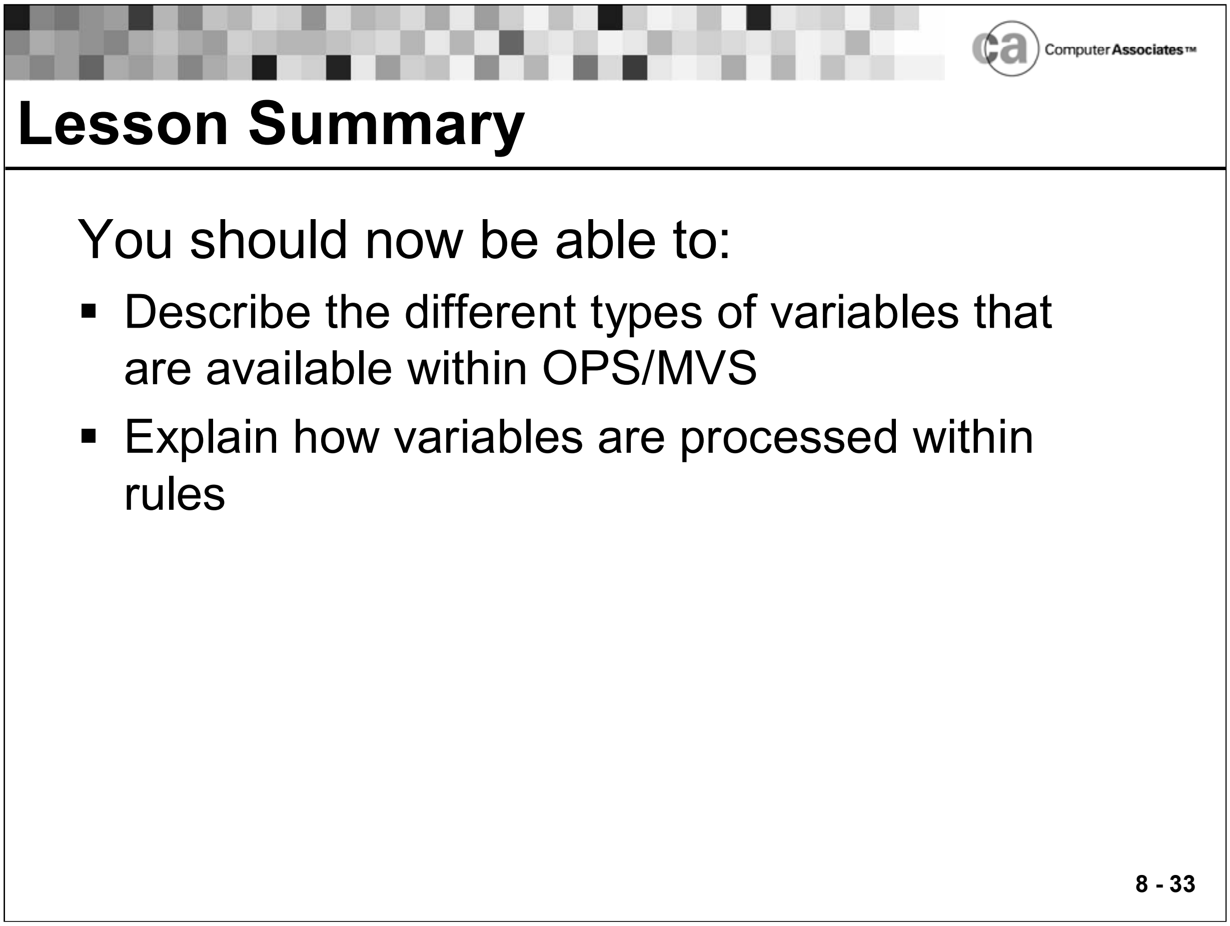
- Rule 3 (within the OPERATNS rule set)

```
)MSG $HASP*
)PROC
/****************************************************************/
/* Rule Purpose: Operations makes the final call on message     */
/*               disposition based on the value of a temporary  */
/*               variable that different groups can override    */
/*               at their request.  Assume disposition is       */
/*               is NORMAL if a group hasn’t set the variable   */
/****************************************************************/
IF OPSVALUE('GLVEVENT.DISP','E') = 'N' THEN DISP = 'NORMAL'
       ELSE DISP = GLVEVENT.DISP
RETURN DISP
```

Temporary Variable Example

The above rule assumes the disposition of the rule is normal when a group has not set the variable.

Rule 1 and Rule 2 execute on a specific $HASP100 event and execute before Rule 3, which has a generic mask of $HASP*. Rule 1 and Rule 2 set the temporary variable GLVEVENT.DISP accordingly. Rule 3 then interrogates the GLVEVENT.DISP variable to determine what action it should take.



Lesson Summary

You should now be able to:

- Describe the different types of variables that are available within OPS/MVS
- Explain how variables are processed within rules

Lesson 8 Assessment

8 - 34

Lesson 8 Assessment

1. All variables have which characteristics?
   a. They can contain character strings of up to 256 bytes.
   b. They are not used in rules.
   c. Derived names of variables can contain up to 50 characters.
   d. Their values can change while a program is running.
   e. b and d.
   f. a, c, and d.

2. Rules can use which type of variable?
   a. Global.
   b. Static.
   c. Local.
   d. Event-related.
   e. Temporary.
   f. All of the above.

Lesson 8 Assessment (continued)

3. Global variables have which characteristics?
   a. They permit serial access.
   b. They have the form GLVTEMPx.XXXX.
   c. They have the form GLOBALx.XXXX.
   d. They allow data to be shared between different rules for events that that occur in the same address space.
   e. b and c.
   f. a, b, and c.
   g. b, c, and d.

4. Static variables have which characteristics?
   a. They maintain a fixed value across multiple executions of a single rule.
   b. They permit serial access.
   c. They are compound symbols.
   d. They are available in the)INIT and)PROC sections of a rule.
   e. a, b, and c.
   f. a and c.
   g. None of the above.

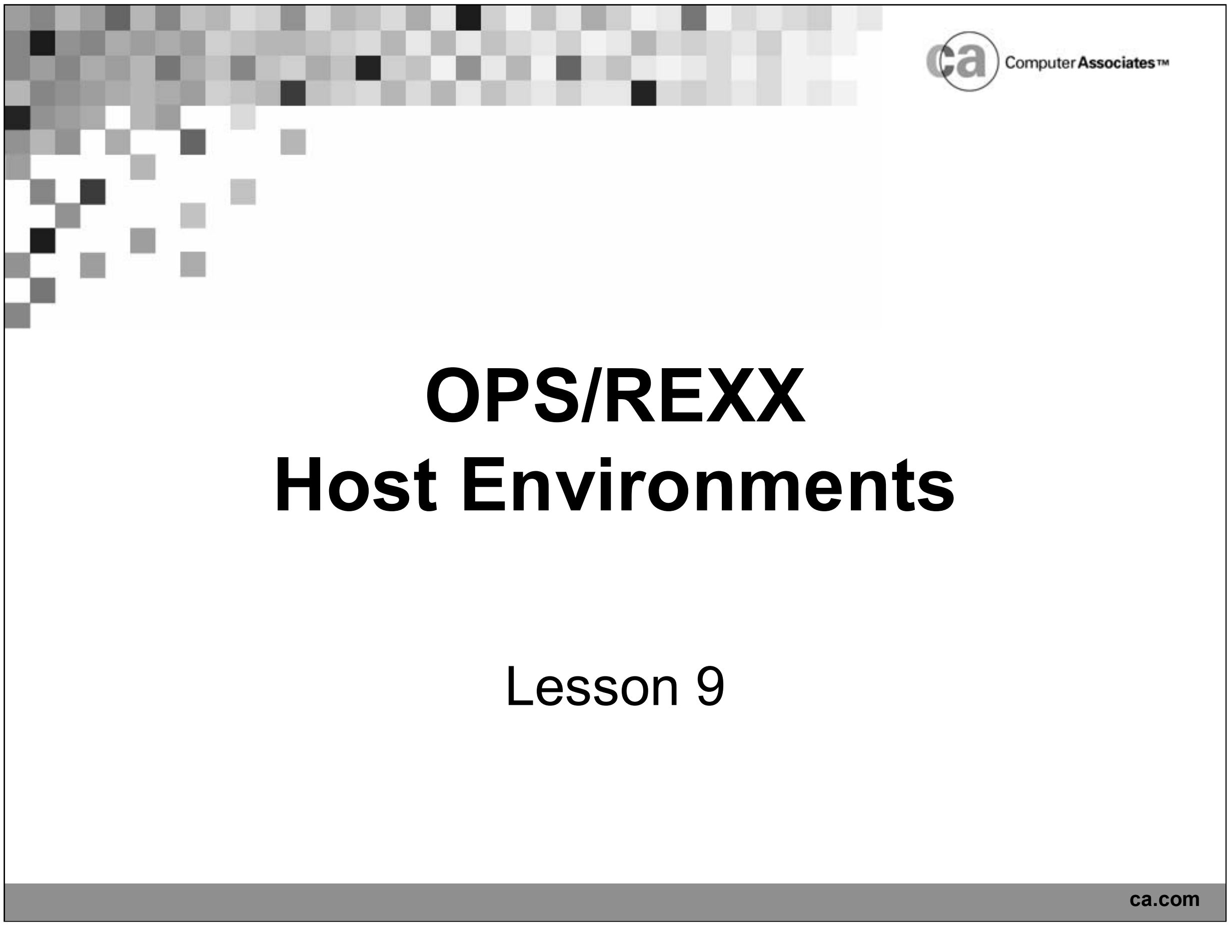
5. Event-related variables have which characteristics?
   a. They are available in the)PROC and)TERM sections of a rule.
   b. They correspond to the rule event types.
   c. They are automatically provided by the AOF engine.
   d. b only.
   e. a, b, and c.
   f. b and c.

Lesson 8 Assessment (continued)

6. Dynamic variables have which characteristics?
   a. Their value can be up to 256 bytes in length.
   b. They are simple variables or non-global compound symbols.
   c. They are created each time a rule executes.
   d. a and c.
   e. a, b, and c.
   f. b and c.
   g. None of the above.

7. Temporary variables have which characteristics?
   a. They have the form GLVTEMP.XXXX.
   b. They have the form GLVEVENT.XXXX.
   c. They allow data to be shared by different rules that are processing the same event.
   d. They are automatically deleted.
   e. They are available only in the)PROC section of a rule.
   f. a, c, and d.
   g. b, c, d, and e.

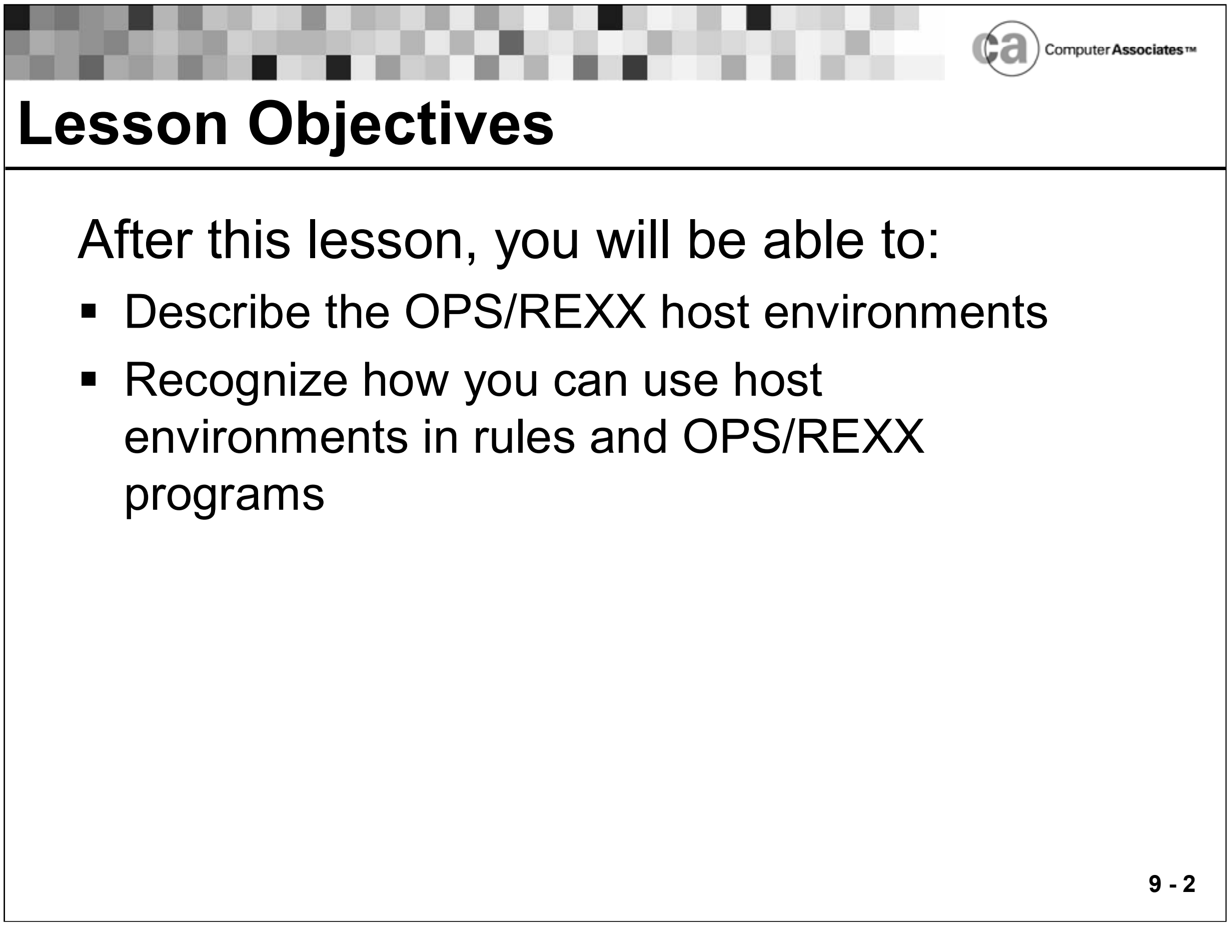
8. Local variables have which characteristics?
   a. They are available only in the)PROC section of a rule.
   b. They are unique to the address space that triggered the rule.
   c. They have the form GLVJOBID.XXXX.
   d. They allow data to be shared between different rules for events that occur in the same address space.
   e. All of the above.



OPS/REXX
Host Environments

Lesson 9

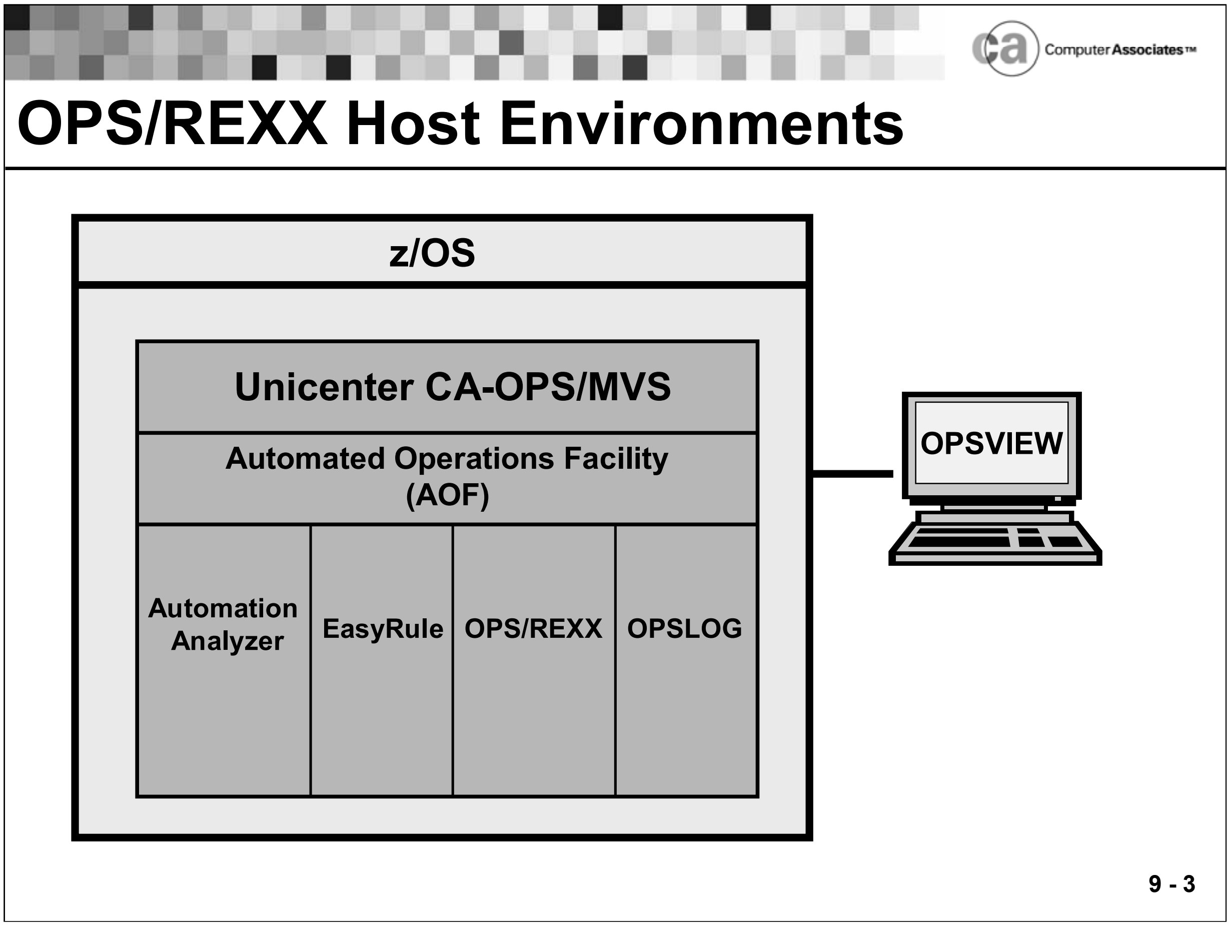
ca.com



Lesson Objectives

After this lesson, you will be able to:

- Describe the OPS/REXX host environments
- Recognize how you can use host environments in rules and OPS/REXX programs



OPS/REXX Host Environments

z/OS

Unicenter CA-OPS/MVS

Automated Operations Facility (AOF)

Automation Analyzer

EasyRule

OPS/REXX

OPSLOG

OPSVIEW

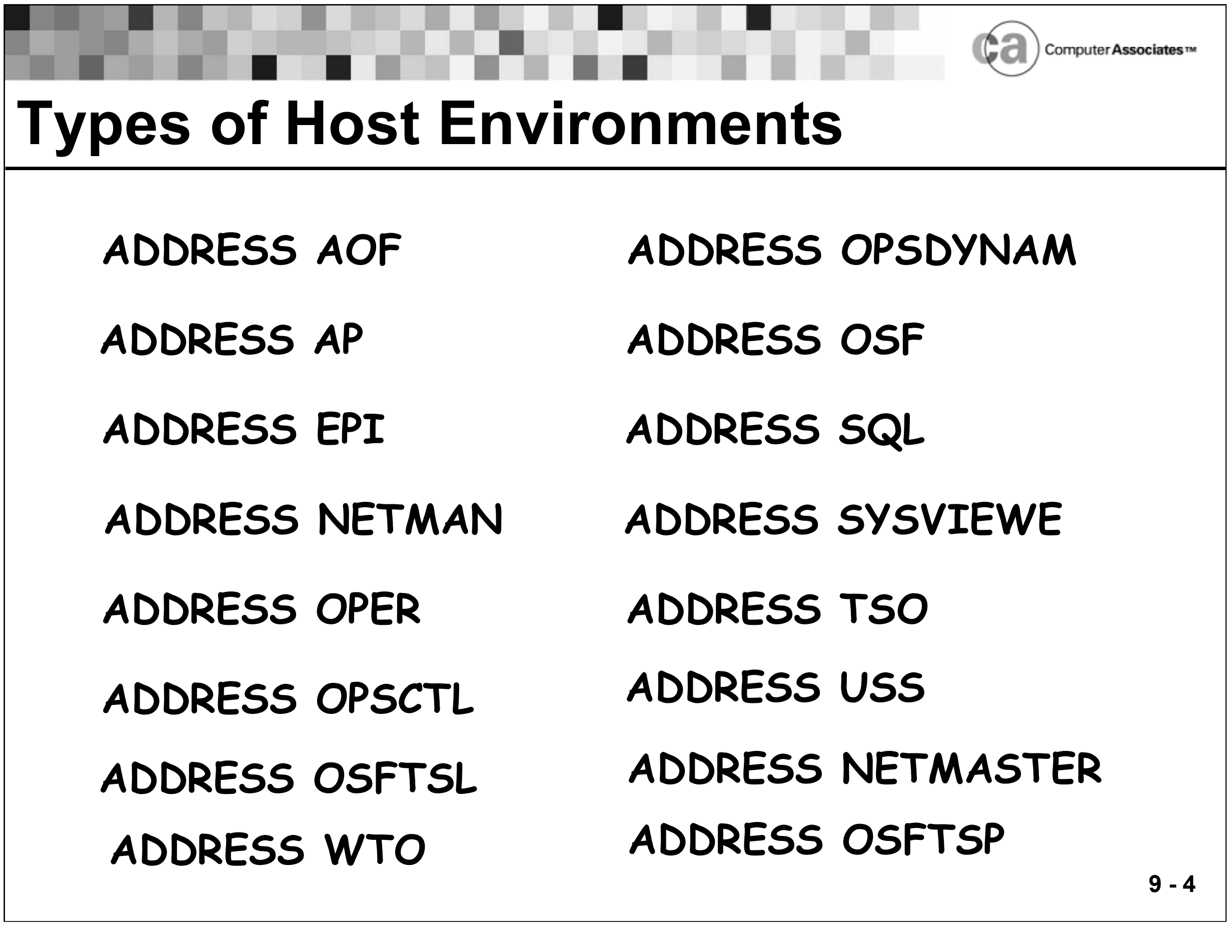
9 - 3

OPS/REXX Host Environments

OPS/REXX host environments are special environments that allow you to control Unicenter CA-OPS/MVS facilities and perform various z/OS tasks. Most host environments are available from AOF rules and OPS/REXX programs. You should always use the host environments rather than their equivalent TSO command processors.

The output that is generated by a host environment depends on the command and the environment from which it was issued.

A host command is a command that an OPS/REXX program or an AOF rule sends to a non-REXX environment for execution. You use an “ADDRESS” instruction in a rule or OPS/REXX program to pass the command to one of the available host environments. See your documentation for more information about these commands.



Types of Host Environments

ADDRESS AOF

ADDRESS AP

ADDRESS EPI

ADDRESS NETMAN

ADDRESS OPER

ADDRESS OPSCTL

ADDRESS OSFTSL

ADDRESS WTO

ADDRESS OPSDYNAM

ADDRESS OSF

ADDRESS SQL

ADDRESS SYSVIEWE

ADDRESS TSO

ADDRESS USS

ADDRESS NETMASTER

ADDRESS OSFTSP

9 - 4

Types of Host Environments

- ADDRESS AOF

  Allows you to programmatically control rules and create dynamic AOF rules.

- ADDRESS AP

  Allows Unicenter CA-OPS/MVS to communicate with an MSF-connected Automation Point system.

- ADDRESS EPI

  Allows OPS/REXX programs to define and operate virtual terminals that interact with VTAM applications via the EPI.

- ADDRESS NETMAN

  Enables you to open, update, or close records in CA-Netman.

- ADDRESS OPER

  Permits you to issue operator commands (z/OS, JES2, JES3, VM) from an OPS/REXX program or AOF rule.

- ADDRESS OPSCTL

  Enables you to control Unicenter CA-OPS/MVS COF, ECF, OSF, and MSF components.

- ADDRESS OPSDYNAM

  Allows you to issue the ADDRESS OPSDYNAM commands for dynamic allocation, concatenation, deconcatenation, and information retrieval functions.

- ADDRESS OSF

  Dispatches an OPS/REXX program to a CA-OPS/MVS TSO address space.

- ADDRESS SQL

  Enables you to create and maintain relational tables.

- ADDRESS SYSVIEWE

  Allows you to send commands to the Unicenter CA-SYSVIEW.

- ADDRESS TSO

  Enables you to route commands to TSO.

- ADDRESS USS

  Allows you to send UNIX or Unicenter TNG Framework API commands to UNIX System Services servers.

- ADDRESS WTO

  Enables you to issue system messages in the form of single line and multi line WTOs and WTORs.

- ADDRESS NETMASTER

  Create, Update, Replace, Close alerts on Unicenter Netmaster Alert Monitor screen

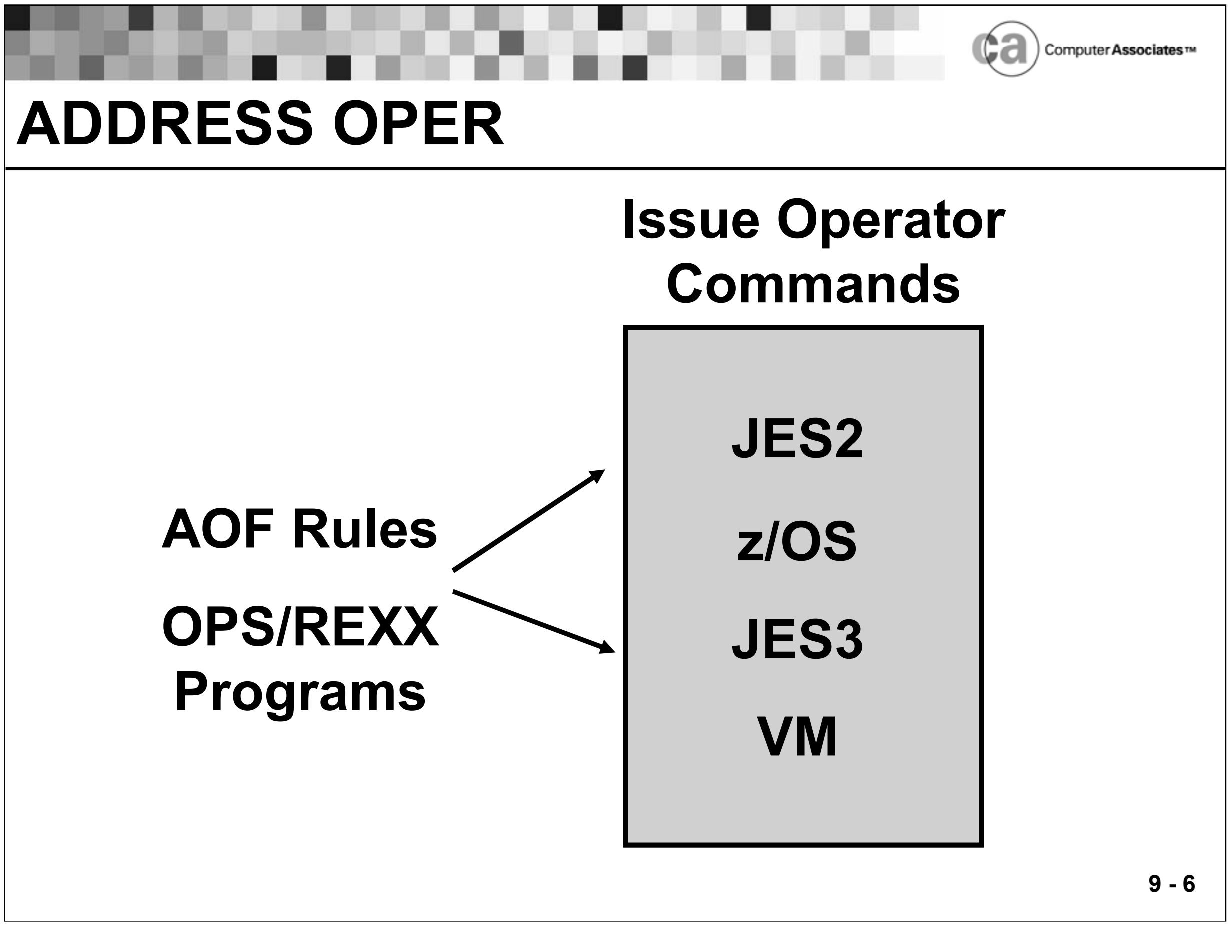
- ADDRESS OSFTSL

  Dispatch a long running OPS/REXX program to a CA-OPS/MVS TSL server address space.

- ADDRESS OSFTSP

  Dispatch a priority OPS/REXX program to a CA-OPS/MVS TSP address space.

Refer to your Unicenter CA-OPS/MVS documentation for specific information about the host environments. This lesson covers the ADDRESS OPER, ADDRESS OSF, and ADDRESS WTO environments in detail.



ADDRESS OPER

Issue Operator Commands

AOF Rules

OPS/REXX Programs

JES2

z/OS

JES3

VM

9 - 6

ADDRESS OPER

The ADDRESS OPER host environment permits you to issue operator commands from an OPS/REXX program or AOF rule. You may use ADDRESS OPER to issue these types of operator commands:

- JES

  Prefix JES2 commands with a dollar sign ($). Prefix JES3 commands with an asterisk (*).

- z/OS (or JES3)

  There is a limit to the number of output lines Unicenter CA-OPS/MVS can capture, but it is so large that truncation is unlikely

- VM

  To issue VM commands, use an ADDRESS OPER instruction and prefix the command with the text □#CP□. Unicenter CA-OPS/MVS issues the command via the standard VM XA/SP Diagnose interface, which requires a buffer to be passed to it. The response to the VM command is returned to this buffer, which must be contiguous in real memory (as seen by z/OS). Because z/OS does not allocate contiguous real memory, VM command responses are currently limited to 4 K, the size of a single real storage frame.

- Use either of the following formats when using the ADDRESS OPER host command environment. Use format 1 when you do not want to specify any additional keywords.
- Format 1

  You may use this format for ADDRESS OPER commands:

  ```
  ADDRESS OPER “command text"
  ```

- Format 2

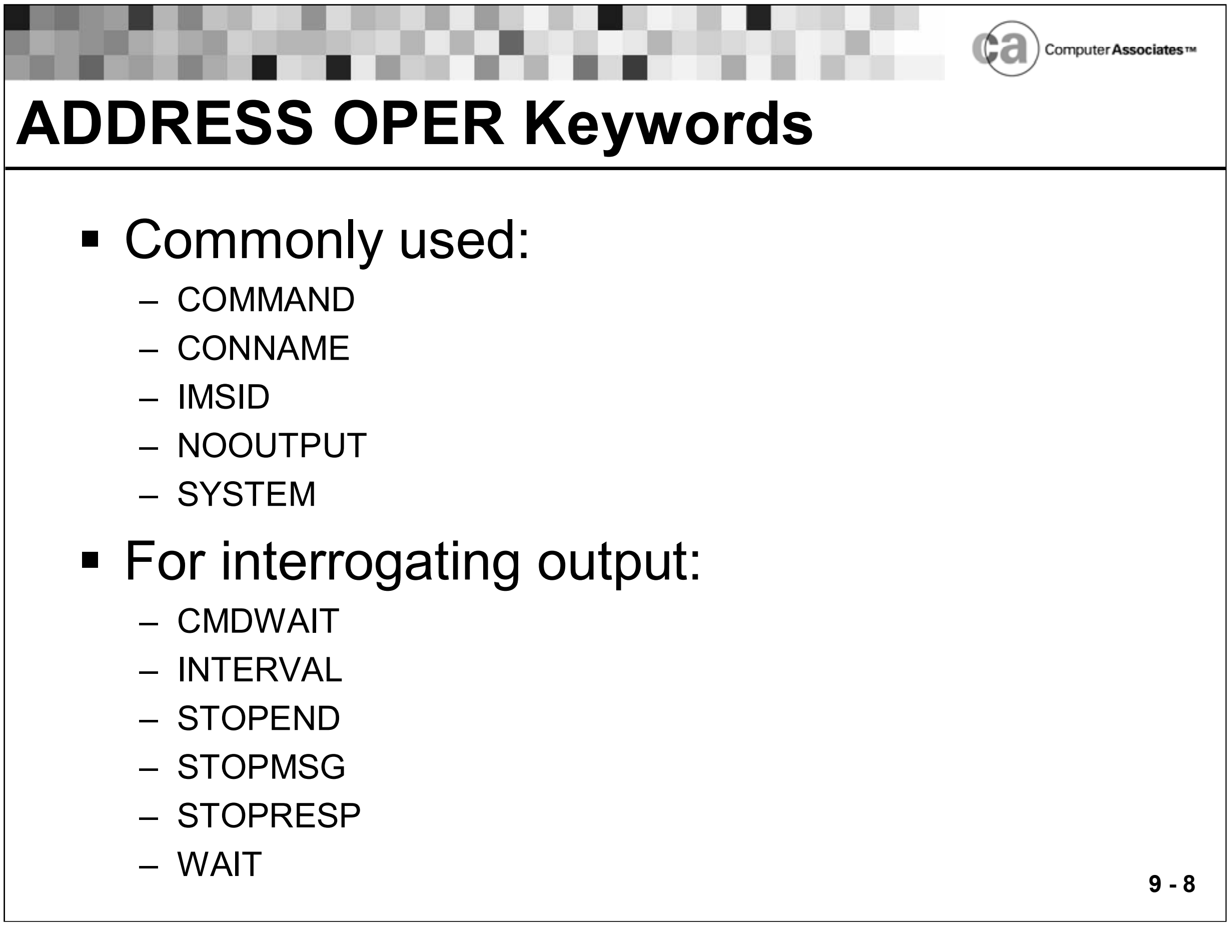
  You may also use the following format for ADDRESS OPER commands:

  ```
  ADDRESS OPER "keywords"
  ```

  If you use this format, the COMMAND keyword must precede any other keywords you specify.

  Optional Keywords

  COMMAND(text)
  CAPTURE(msgtextlist)
  CMDECHO(YES|NO)
  CMDLOG(YES|NO)
  CMDWAIT(seconds)
  CONTYPE(ANY|EXTCONS|MIGCONS|SSCONS)
  DELAY(seconds)
  ID|CONID(consoleid)
  IMSID(imsid)
  INTERVAL(centiseconds)
  LOG(YES|NO|OFF)
  MAXCMDOUT(number)
  NAME|CONNAME(consolename)
  NOCLIST
  OUTPUT|NOOUTPUT
  STOPEND(YES|NO)
  STOPMSG(msgtextlist)
  STOPRESP(msgtextlist)
  SYSID|SYSTEM(systemids)
  SYSWAIT(seconds)
  WAIT(seconds)



ADDRESS OPER Keywords

- Commonly used:
  - COMMAND
  - CONNAME
  - IMSID
  - NOOUTPUT
  - SYSTEM
- For interrogating output:
  - CMDWAIT
  - INTERVAL
  - STOPEND
  - STOPMSG
  - STOPRESP
  - WAIT

9 - 8

ADDRESS OPER

The following keywords are some of the most commonly used keywords in the ADDRESS OPER environment:

- COMMAND

  The text of the operator command you wish to issue.
- CONNAME

  The name of the console that is to receive the command.
- IMSID

  The name of the IMS control region that is to receive the command.
- NOOUTPUT

  Requests that no command output be returned.
- SYSTEM

  The name of the system to which you want to route the command.

The following keywords are used to interrogate output in the ADDRESS OPER environment:

- CMDWAIT

  Specifies (in seconds) how long the OPSCMD command processor should wait for command output collection to complete.

- INTERVAL

  Specifies (in centi-seconds) how long the OPSCMD command processor waits before testing the response lines to see if the response has ended.

- STOPEND

  Determines whether the end line of a multi-line WTO message stops the OPSCMD command processor from collecting further command output.

- STOPMSG

  Specifies a list of one to ten message text segments that terminate the collection of command response lines. The message segment(s) you specify do not need to be directed to the console receiving the command response. STOPMSG and STOPRESP are mutually exclusive.

- STOPRESP

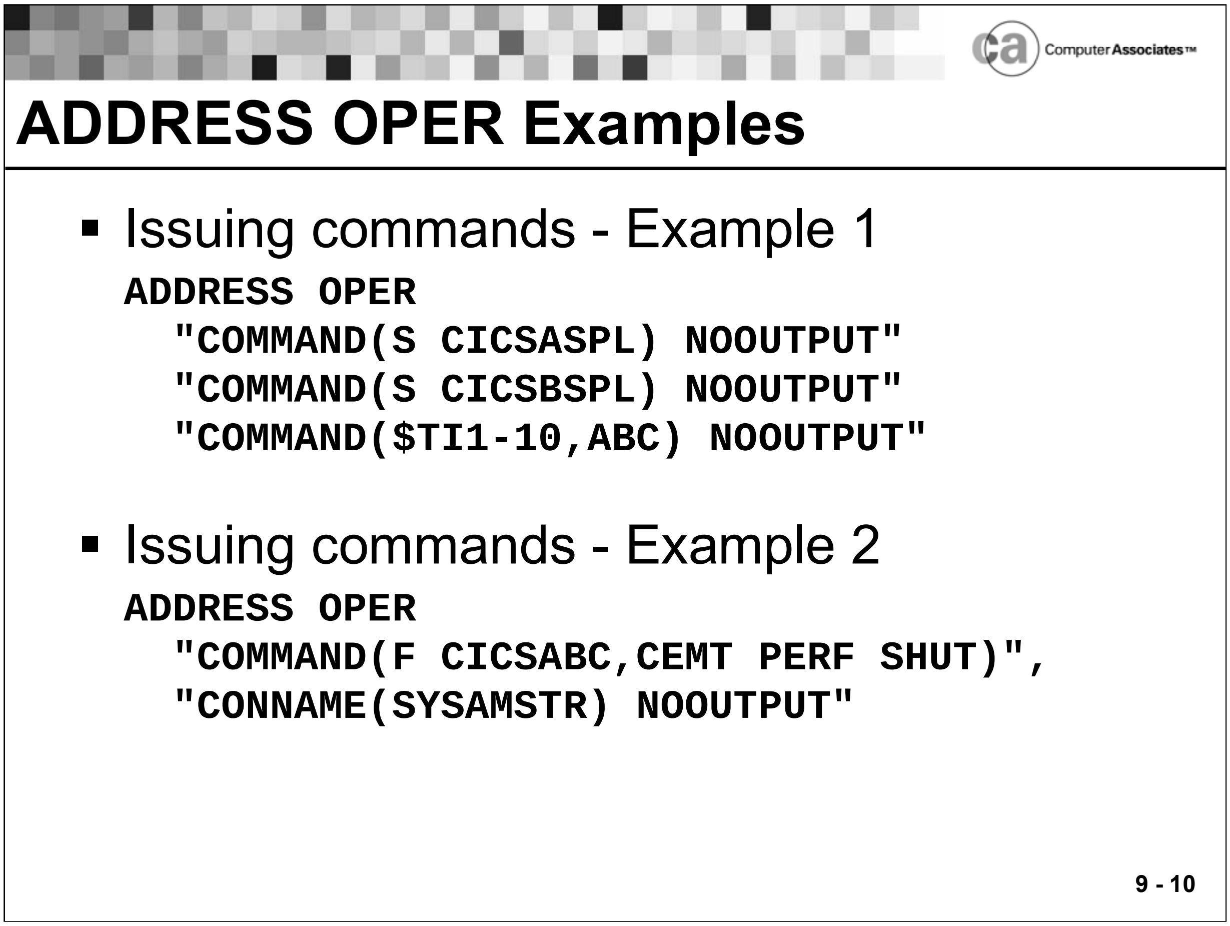
  Specifies a list of one to ten message text segments that terminate the collection of command response lines. The message segment(s) you specify must be directed to the console receiving the command response. STOPMSG and STOPRESP are mutually exclusive.

- WAIT

  Specifies how long the OPSCMD command processor waits, unconditionally, to receive all output from the current command.

Special Note:

- You can only interrogate command responses from commands issued via the ADDRESS OPER host environment within AOF request (REQ) and time-of-day (TOD) rules.
- Generally, no waiting is allowed in AOF rules (except request and time-of-day rules); therefore, you cannot collect command responses because doing so would suspend the address space in which the AOF is processing. If you need to interrogate the output from a command (for which no OPS/REXX function exists) for an AOF rule in which no waiting is allowed, you should trigger an OPS/REXX program to an OPSOSF server.



ADDRESS OPER Examples

- Issuing commands - Example 1

```
ADDRESS OPER
  "COMMAND(S CICSASPL) NOOUTPUT"
  "COMMAND(S CICSBSPL) NOOUTPUT"
  "COMMAND($TI1-10,ABC) NOOUTPUT"
```

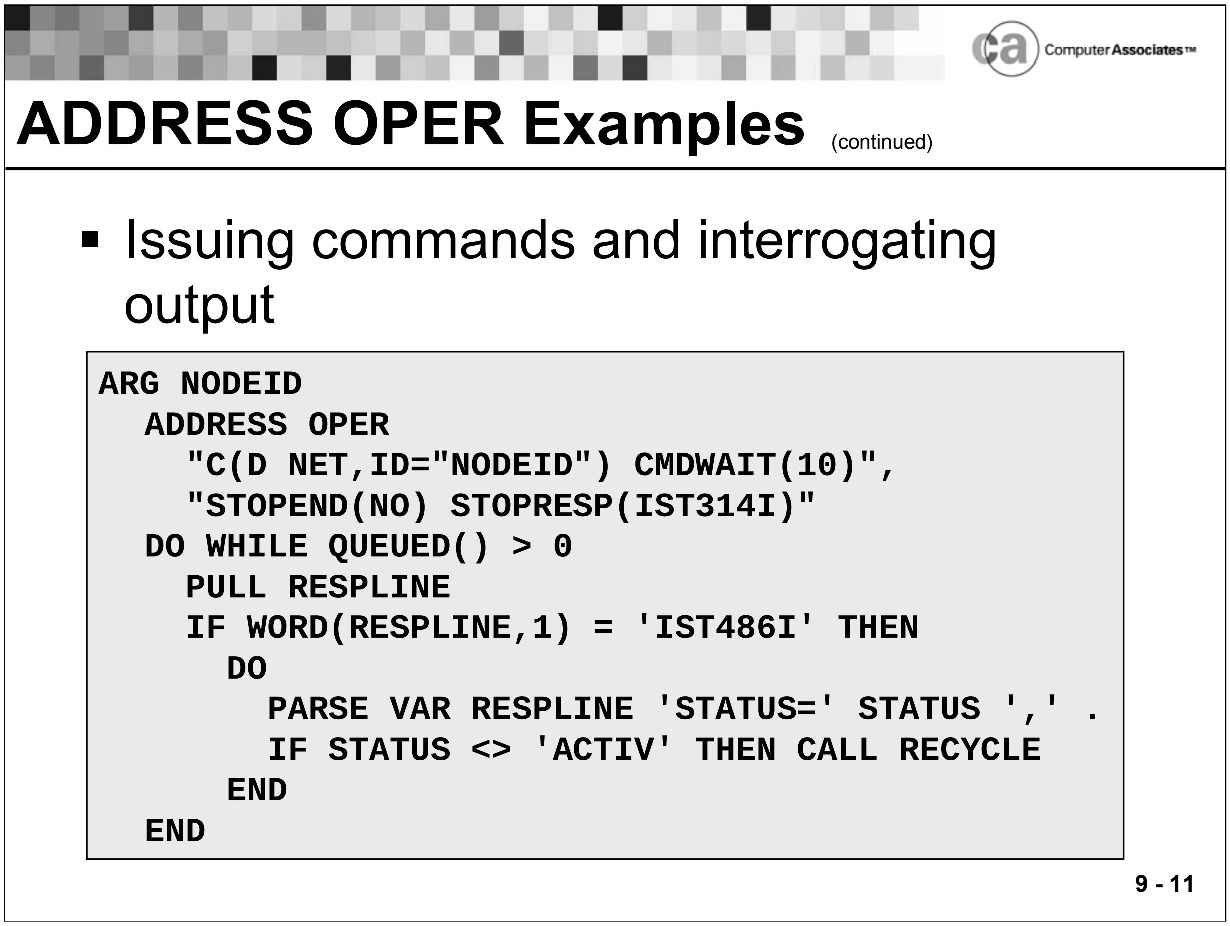
- Issuing commands - Example 2

```
ADDRESS OPER
  "COMMAND(F CICSABC,CEMT PERF SHUT)",
  "CONNAME(SYSAMSTR) NOOUTPUT"
```

ADDRESS OPER Examples

Example 1 illustrates issuing a series of z/OS commands.

Example 2 uses ADDRESS OPER keywords (COMMAND, CONNAME) to cause a command to be issued on behalf of a particular console.



ADDRESS OPER Examples (continued)

- Issuing commands and interrogating output

```
ARG NODEID
  ADDRESS OPER
    "C(D NET,ID="NODEID") CMDWAIT(10)",
    "STOPEND(NO) STOPRESP(IST314I)"
  DO WHILE QUEUED() > 0
    PULL RESPLINE
    IF WORD(RESPLINE,1) = 'IST486I' THEN
      DO
        PARSE VAR RESPLINE 'STATUS=' STATUS ',' .
        IF STATUS <> 'ACTIV' THEN CALL RECYCLE
      END
  END
```

ADDRESS OPER Examples

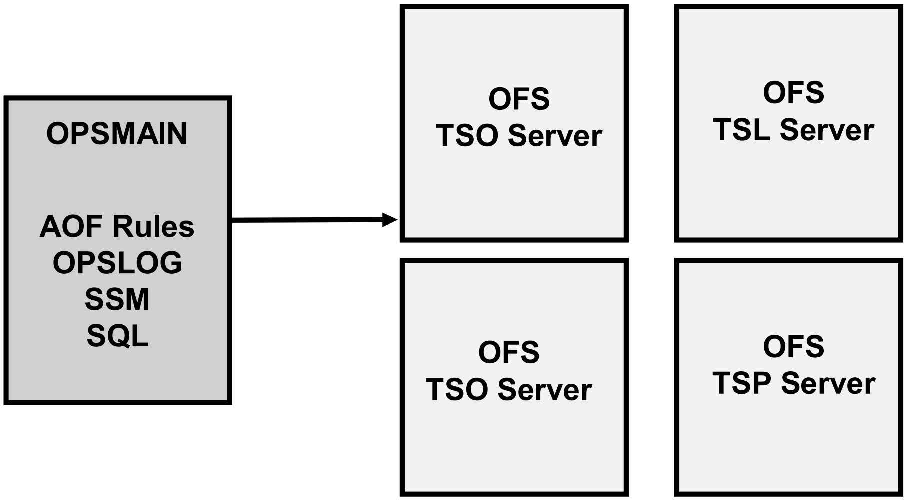
The above example utilizes ADDRESS OPER keywords (CMDWAIT, STOPEND, STOPRESP) to issue commands and interrogate output.

Notes:

- Except for)TOD and)REQ rules, output cannot be interrogated from rules.
- Warning! Although output can be interrogated with)TOD rules, it is not recommended to do so.
- Output is returned to the external data queue.

ADDRESS OSF/OSFTSL/OSFTSP

Dispatch programs to OSF Servers



OPSMAIN

AOF Rules
OPSLOG
SSM
SQL

OFS
TSO Server

OFS
TSL Server

OFS
TSO Server

OFS
TSP Server

9 - 12

ADDRESS OSF/OSFTSL/OSFTSP

One of Unicenter CA-OPS/MVS's most powerful features is its ability to process system events inline or synchronously via the AOF (rules) engine. This type of real-time automation is extremely effective and is possible because AOF rules execute in the address space in which an event occurs. Most of your automation efforts such as issuing WTOs, suppressing messages, querying asids, devices or initiators, issuing OS/390 or JES2 commands (no command output interrogation needed), manipulating global variables, manipulating RDF tables, etc, can therefore be effectively accomplished within AOF rules.

The types of automated logic that you will not be able to perform within rules are those that require some interaction or any type of wait. These types of interaction or waiting automation include:

- Issuing OS/390, JES2 or JES3, USS, and VM commands (in which no OPS/REXX function exists) AND interrogation of the command response is needed.
- Issuing WTORs AND interrogating the reply that was given.
- Utilizing ISPF services (ADDRESS ISPEXEC)
- File I/O manipulation (allocating and reading/writing of files)
- Use of the OPSWAIT()
- Use of the EPI and SYSVIEWE OPS/REXX Host Environments

ADDRESS OSF/OSFTSL/OSFTSP (continued)

These types of automation need to be performed by dispatching an OPS/REXX program to an OPS/MVS OSF server via the ADDRESS OSF, ADDRESS OSFTSL, or ADDRESS OSFTSP host environment.

The CA-OPS/MVS OSF servers are separate TSO (IKJEFT01) address spaces that are started by the CA-OPS/MVS OPSMAIN address space. The servers allocate the data set (or data sets) that will contain the OPS/REXX programs that are dispatched. Various OSFx, OSFTSLx, and OSFTSPx, related CA-OPS/MVS parameters determine the types and the minimum/maximum number of servers to start, as well parameters that regulate the CPU and time usage of the programs that execute within them. The type of automation created within each site will determine the settings of these parameters.

The TSL and TSP servers are also TSO servers, but are assigned special execution classifications. The TSL servers are servers in which long running OPS/REXX programs can be dispatched. The TSP servers are servers in which priority type automated logic can be dispatched to guarantee its immediate execution.

Most dispatched OPS/REXX programs from AOF rules to servers will be done via the ADDRESS OSF host environment: (with OSFx parameters defined)

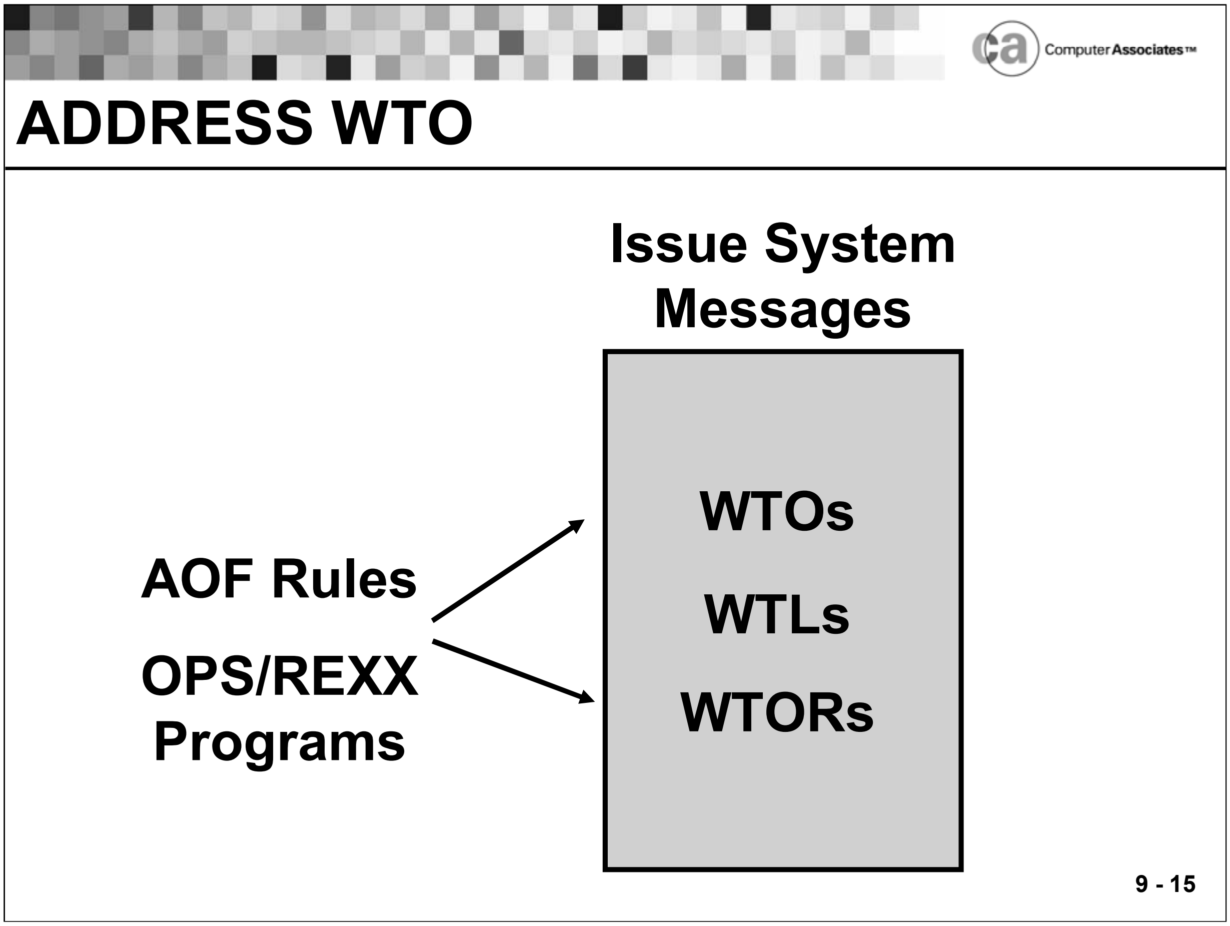
```
Address OSF
“OI PROGRAM(program)”
```

An OPS/REXX program that may contain lengthy waiting can be triggered to a TSL server via the ADDRESS OSFTSL host environment:(with OSFTSLx parameters defined)

```
Address OSFTSL
“OI PROGRAM(program)”
```

An OPS/REXX program that must run immediately (performs some type of system health check logic) can be dispatched to a priority TSP server via the ADDRESS OSFTSP host environment:

```
Address OSFTSP
“OI PROGRAM(program)”
```



ADDRESS WTO

Issue System Messages

AOF Rules

OPS/REXX Programs

WTOs

WTLs

WTORs

9 - 15

ADDRESS WTO

The ADDRESS WTO host environment enables you to issue system messages in the form of WTOs, WTLs, or WTORs. Using this host environment, you can:

- Issue a WTO message synchronously in AOF rules, causing the WTO to be sent when the rule executes. When you send a WTO message by using the OPSWTO command processor in a rule, the OPSWTO command goes to an OSF server. This can cause a slight delay between the rule executing and the WTO being issued.
- Issue multiline WTO messages. The ADDRESS WTO host environment uses REXX stem variables to store individual lines of the message.

To issue a single-line WTO message:

```
ADDRESS WTO "TEXT('messagetext') keywords”
```

- Optional Keywords:

  AREAID(areaid)
  CNID(consoleids)
  CNNAME(consolenames)
  DELAY(delaytime)
  DESC(desccode)
  HILITE|LOWLITE
  MCSFLAGS(flagvalues)
  MSGID(messageid)
  OPTION(value)
  REPLY
  ROUTE(routecode)
  SUBSYS(ssid)
  SYSTEM(ALL|EXT|sysnames)
  SYSWAIT(seconds)
  TOKEN(dom token)
  WAIT(waittime)
  WTOID(wtoid)

Note: Specifying a wtoid value for the WTOID keyword is optional.

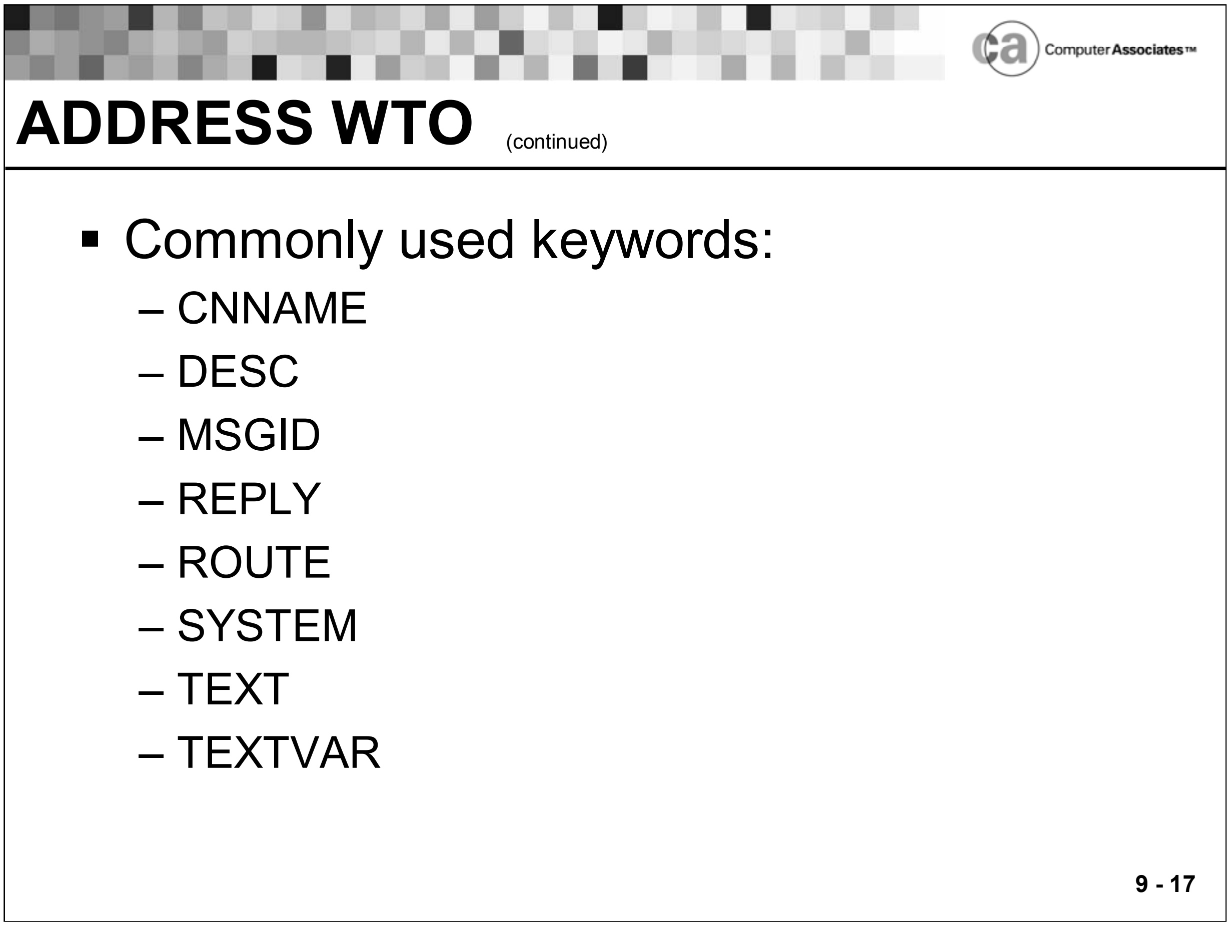
To issue a multiline WTO message:

```
ADDRESS WTO "TEXTVAR(stem-name) keywords”
```

- Keywords

  (same as previous format)

For detailed information about these keywords, see your Unicenter CA-OPS/MVS documentation. This lesson focuses on the most commonly used keywords.



ADDRESS WTO (continued)

- Commonly used keywords:
  - – CNNAME
  - – DESC
  - – MSGID
  - – REPLY
  - – ROUTE
  - – SYSTEM
  - – TEXT
  - – TEXTVAR

9 - 17

ADDRESS WTO

The following keywords are some of the most commonly used keywords in the ADDRESS WTO environment:

- CNNAME

  The name(s) of the console(s) that are to receive the WTO or WTOR message. (You can specify up to 16 alphanumeric names containing up to eight characters each.)

- DESC

  The message descriptor codes for the current message.

- MSGID

  The message ID that prefixes the WTO text.

- REPLY

  The reply for the message.

- ROUTE

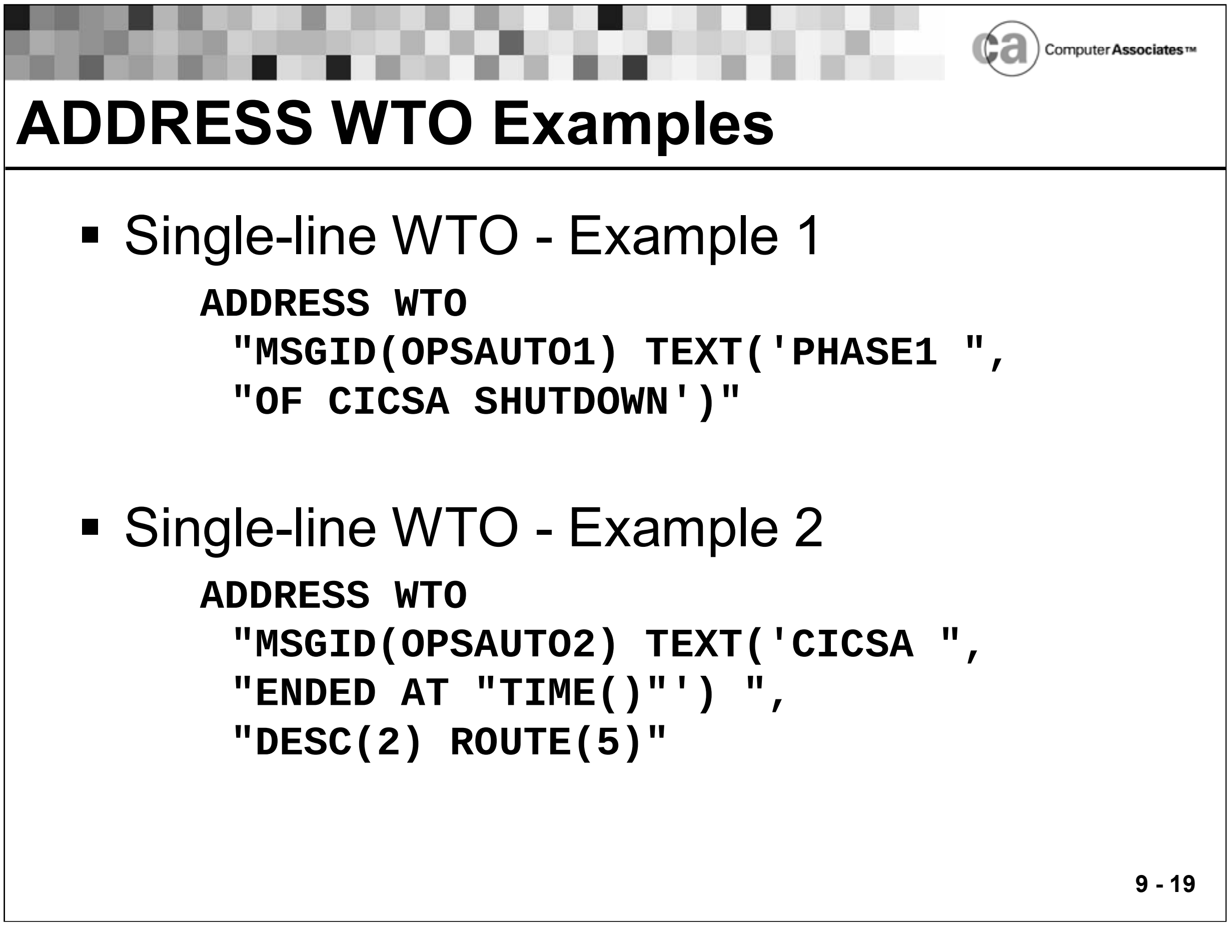
  The route codes for the message.

- SYSTEM

  The name of the system that is to receive the WTO message.
- TEXT

  The text of a single-line WTO message.
- TEXTVAR

  The text of a multi-line WTO message.



ADDRESS WTO Examples

- Single-line WTO - Example 1

```
ADDRESS WTO
 "MSGID(OPSAUTO1) TEXT('PHASE1 ",
 "OF CICSA SHUTDOWN')"
```

- Single-line WTO - Example 2

```
ADDRESS WTO
 "MSGID(OPSAUTO2) TEXT('CICSA ",
 "ENDED AT "TIME()"') ",
 "DESC(2) ROUTE(5)"
```

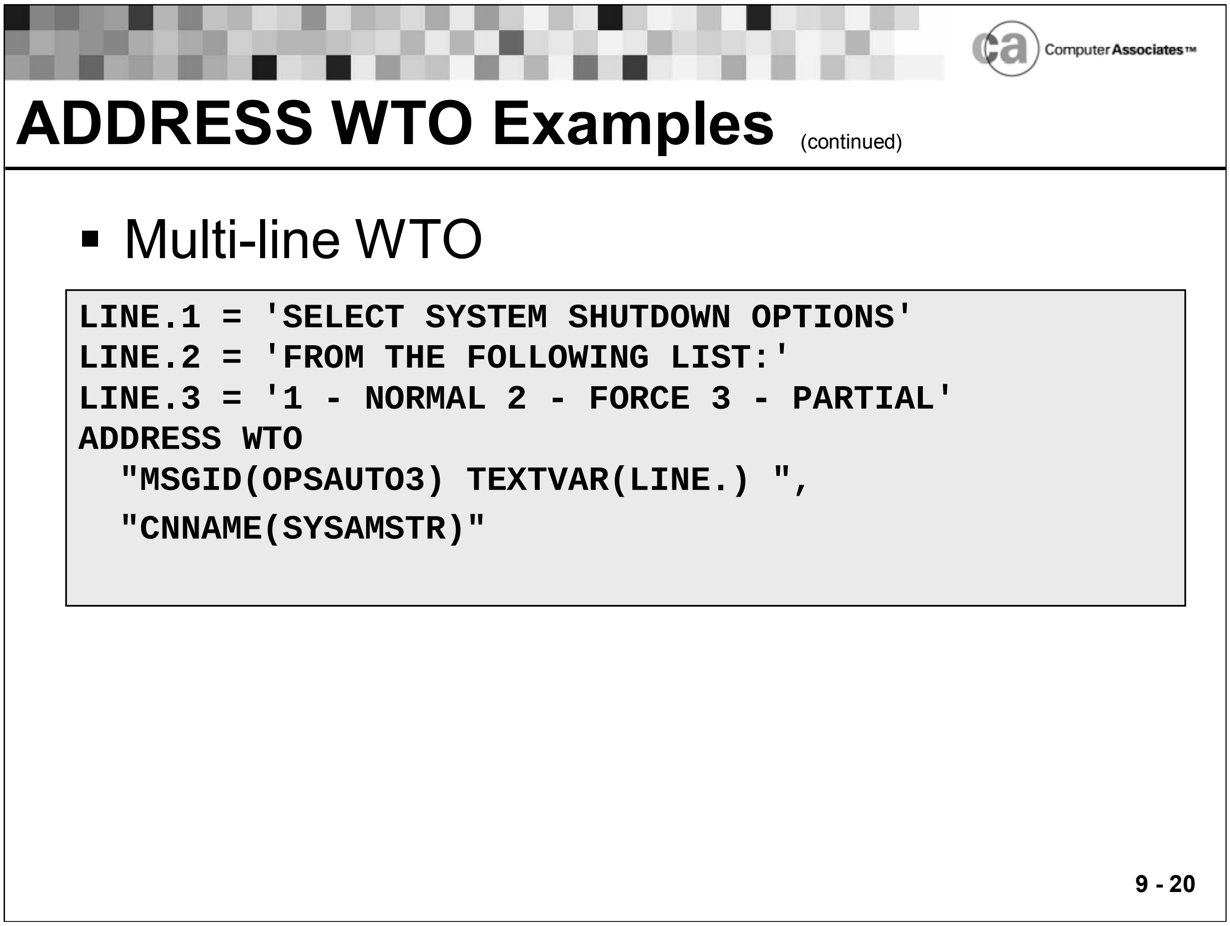
9 - 19

ADDRESS WTO Examples

Example 1 issues the “OPSAUTO1 PHASE1 OF CICSA SHUTDOWN” message to the system.

Example 2 issues the highlighted “OPSAUTO2 CICSA ENDED AT TIME()” message to the system.

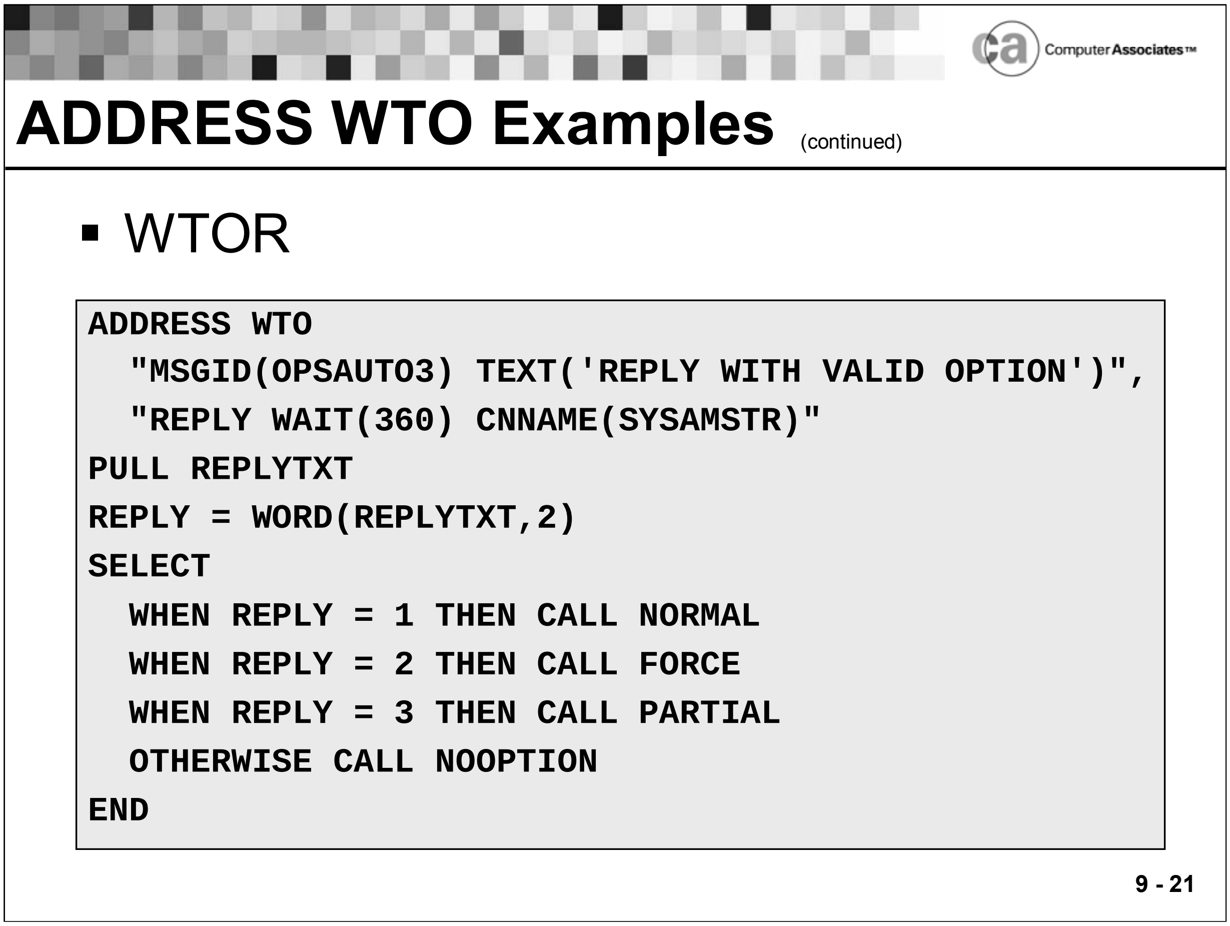
Note: If ROUTE() is omitted, the WTODEFAULTROUTE parameter or the DEFAULT statement of the CONSOLxx member sets the route code.



ADDRESS WTO Examples (continued)

- Multi-line WTO

```
LINE.1 = 'SELECT SYSTEM SHUTDOWN OPTIONS'
LINE.2 = 'FROM THE FOLLOWING LIST:'
LINE.3 = '1 - NORMAL 2 - FORCE 3 - PARTIAL'
ADDRESS WTO
  "MSGID(OPSAUTO3) TEXTVAR(LINE.) ",
  "CNNAME(SYSAMSTR)"
```

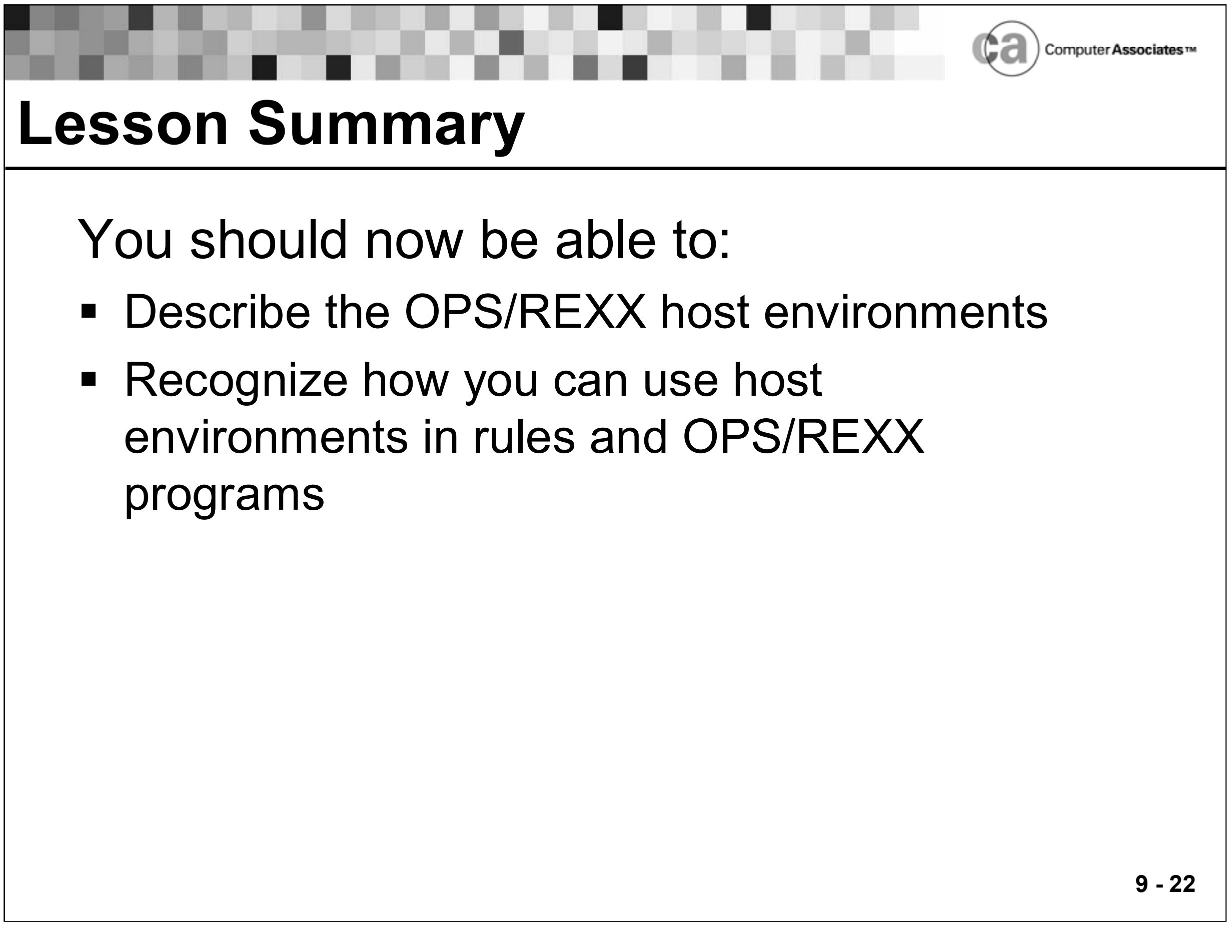


ADDRESS WTO Examples (continued)

- WTOR

```
ADDRESS WTO
  "MSGID(OPSAUTO3) TEXT('REPLY WITH VALID OPTION')",
  "REPLY WAIT(360) CNNAME(SYSAMSTR)"
PULL REPLYTXT
REPLY = WORD(REPLYTXT,2)
SELECT
  WHEN REPLY = 1 THEN CALL NORMAL
  WHEN REPLY = 2 THEN CALL FORCE
  WHEN REPLY = 3 THEN CALL PARTIAL
  OTHERWISE CALL NOOPTION
END
```

9 - 21



Lesson Summary

You should now be able to:

- Describe the OPS/REXX host environments
- Recognize how you can use host environments in rules and OPS/REXX programs

9 - 22

Lesson 9 Assessment

9 - 23

Lesson 9 Assessment

Write the appropriate letter next to each description below.

___ Issue system messages as WTOs and WTORs.

___ Control Unicenter CA-OPS/MVS components.

___ Route commands to TSO.

___ Issue commands for dynamic allocation, concatenation, deconcatenation, and information retrieval functions.

___ Issue operator commands from an OPS/REXX program or AOF rule.

___ Send UNIX commands to servers.

___ Create and maintain relational tables.

___ Programmatically control rules and create dynamic AOF rules.

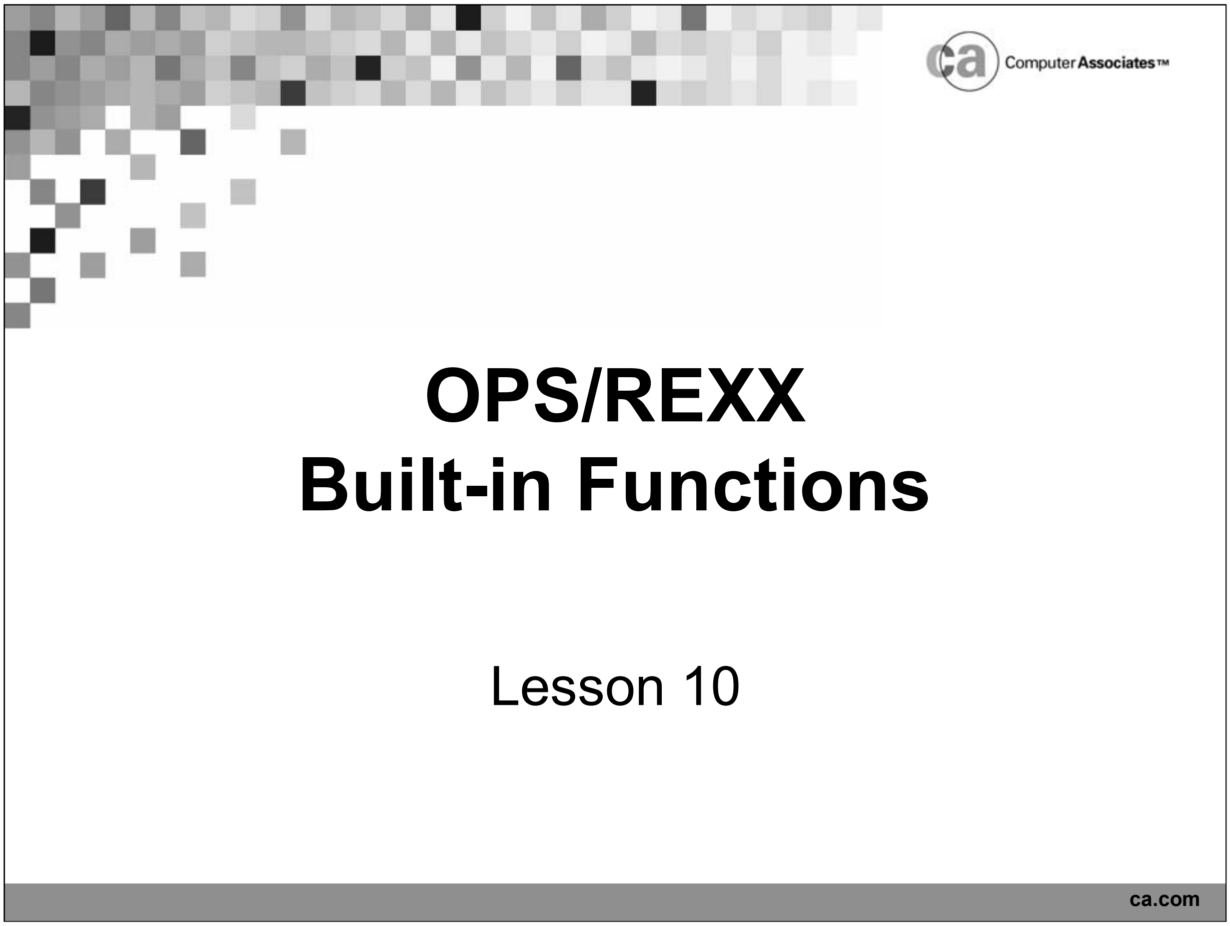
___ Route commands to dispatch an OPS/REXX program to a server.

___ Programmatically control VTAM applications.

___ Dispatch an OPS/REXX program to a specially classified long running server.

___ Dispatch an OPS/REXX program to a specially classified priority server.

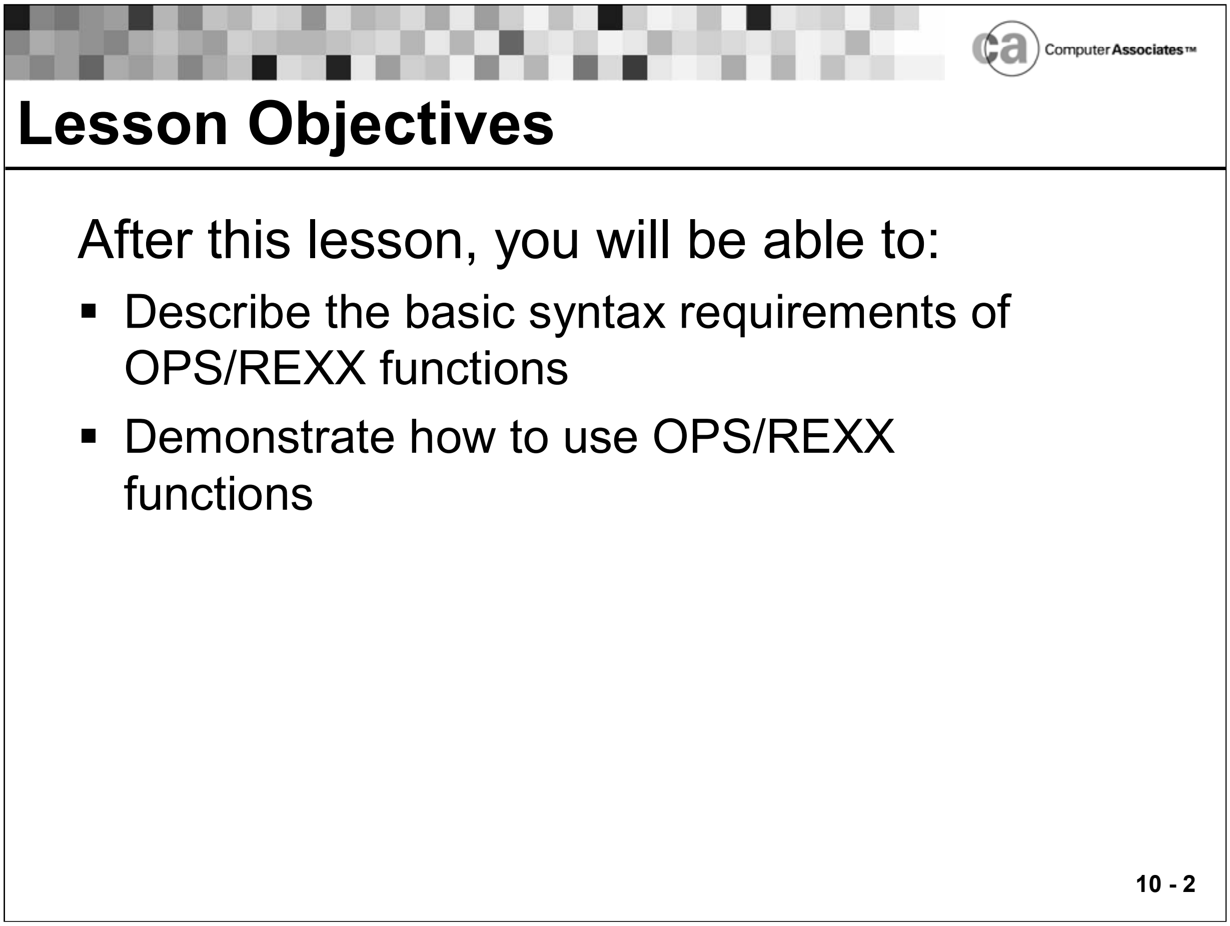
a. ADDRESS USS
b. ADDRESS AOF
c. ADDRESS NETMAN
d. ADDRESS RDF
e. ADDRESS REXX
f. ADDRESS SYSVIEWE
g. ADDRESS OPSCTL
h. ADDRESS WTO
i. ADDRESS MSF
j. ADDRESS OPSDYNAM
k. ADDRESS OPER
l. ADDRESS TSO
m. ADDRESS EPI
n. ADDRESS SQL
o. ADDRESS SYSVIEW
p. ADDRESS OSF
q. ADDRESS OSFTSL
r. ADDRESS OSFTSP



OPS/REXX Built-in Functions

Lesson 10

ca.com

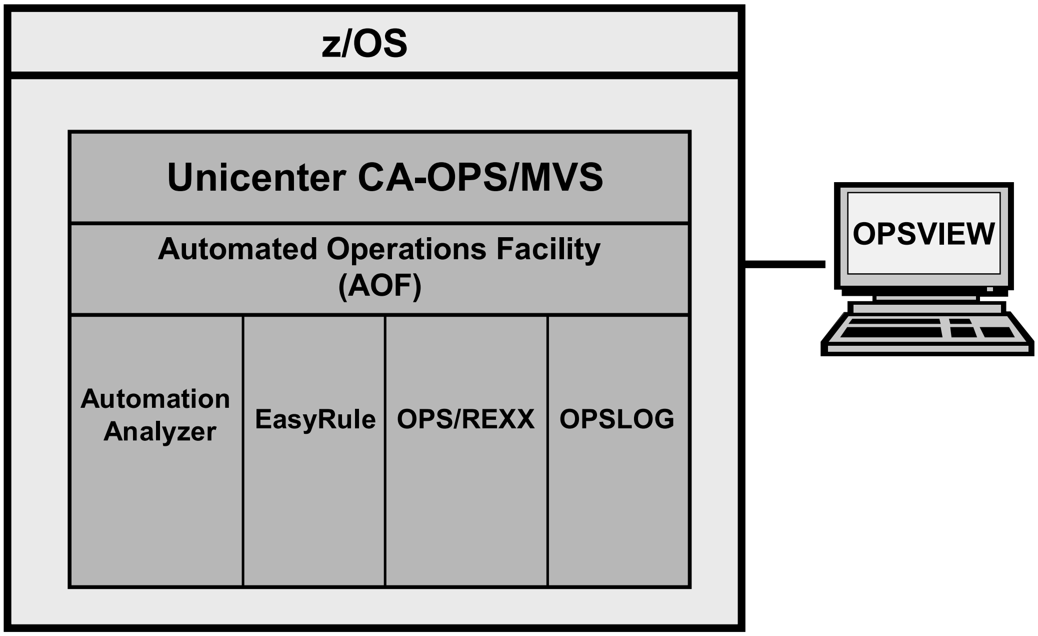


Lesson Objectives

After this lesson, you will be able to:

- Describe the basic syntax requirements of OPS/REXX functions
- Demonstrate how to use OPS/REXX functions

OPS/REXX Functions



10 - 3

OPS/REXX Functions

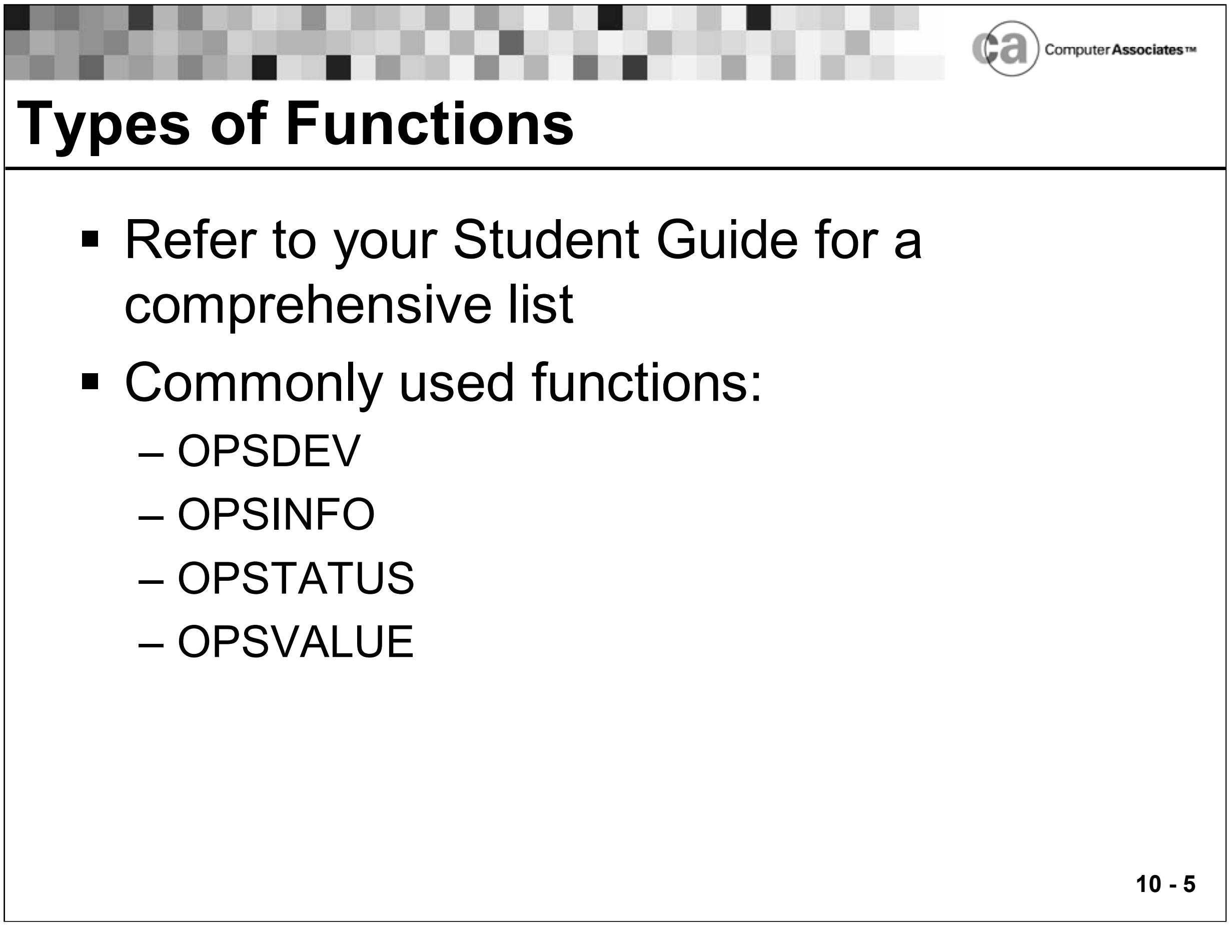
- Built-in functions are an integral component of OPS/REXX that give AOF rules access to a wide variety of system data that is necessary to take programmatic actions and, in some cases, perform specific system tasks.
- OPS/REXX functions can retrieve such system information as job status, device status, and JES2-related resource data. They also allow you to perform tasks like low-lighting messages or performing Automation Restart Management (ARM) services.

A Note About REXX Built-in Functions

- REXX's 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 Unicenter CA-OPS/MVS.
- Built-in functions operate under OPS/REXX exactly as they do under standard REXX, except for the differences described in your Unicenter CA-OPS/MVS documentation.
- For information about standard REXX functions, refer to the second edition of the Cowlishaw book, which you can order from Prentice-Hall.

Functions Not Supported in OPS/REXX

- CHARIN
- CHAROUT
- CHARS
- LINEIN
- LINEOUT
- LINES
- The S (Scan) option of the TRACE function



Types of Functions

- Refer to your Student Guide for a comprehensive list
- Commonly used functions:
  - OPSDEV
  - OPSINFO
  - OPSTATUS
  - OPSVALUE

10 - 5

Available Functions

- OPSARM

  Issue requests for z/OS automatic restart management (ARM) services from AOF rules on behalf of the current address space.

- OPSAUTO

  Query and change the auto-enable status of rule members before z/OS is active.

- OPSBITS

  Return a character string whose internal binary representation is all binary zeros except that the bits its arguments specify are on.

- OPSBN

  Return information about certain keywords that appear in translated CA-AutoMate/MVS rules

- OPSCA7

  Issue CA-7 commands from within your OPS/REXX programs. The OPSCA7 function cannot be used in rules.

- OPSCAWTO

  Send a message to the Unicenter TNG Event Manager in the form of an SNMP trap.

- OPSCLEDQ

  Clear the external data queue.

- OPSCOLOR

  Change the color of the messages in the OPSLOG Browse display.

- OPSCPF

  Obtain information about the z/OS Command prefix Facility (CPF).

- OPSDEV

  Obtain device information synchronously.

- OPSDELV

  Delete global variables whose names match a name mask that you specify and return a count of the deleted variables.

- OPSDOM

  Delete operator messages.

- OPSDUMP

  Tell Unicenter CA-OPS/MVS to take an SVC dump of the current address space. You can use OPSDUMP only in AOF rules.

- OPSECURE

  Return information about the eTrust CA-ACF2, RACF, and eTrust CA-Top Secret security packages.

- OPSENQ

  Interact with z/OS ENQ/DEQ services to serialize use of resources.

- OPSGETV

  Retrieve the value of a global variable that you specify.

- OPSGETVL

  Retrieve the names of global variables that match a variable name mask that you specify.

- OPSHFI

  Read and write variables to a shared VSAM file.

- OPSINFO

  Obtain information about Unicenter CA-MVS and/or the environment where the issuing OPS/REXX program is running.

- OPSIPL

  Obtain various information about the IPL parameter library.

- OPSJES2

  Obtain JES2-related resource data.

- OPSLOG

  Extract messages from the OPSLOG.

- OPSPRM

  Control Unicenter CA-OPS/MVS parameters.

- OPSPRMLB

  Access the z/OS Logical Parmlib Concatenation facility.

- OPSSEND

  Send a message from one copy of Unicenter CA-OPS/MVS to another copy and specify how the receiving copy processes the message. You can also send a message to the OPSLOG of the current copy.

- OPSSETV

  Create a global variable or update the value of a global variable.

- OPSSMF

  Create SMF records.

- OPSSMTBL

  Maintain the directory table that System State Manager uses to manage tables that contain information about system resources.

- OPSSRM

  Obtain information about the z/OS System Resource Manager (SRM).

- OPSTATUS

  Obtain information about the address spaces that are currently in the system. Optionally, you can also write one or more records to the external data queue.

- OPSTORE

  Obtain information about the virtual storage of a specified address.

- OPSUBMIT

  Submit a batch job.

- OPSVALUE

  Manipulate variables.

- OPSVSAM

  Read and write VSAM files.
- OPSYSPLX

  Return information about a system in the current sysplex.
- OPSYSSYM

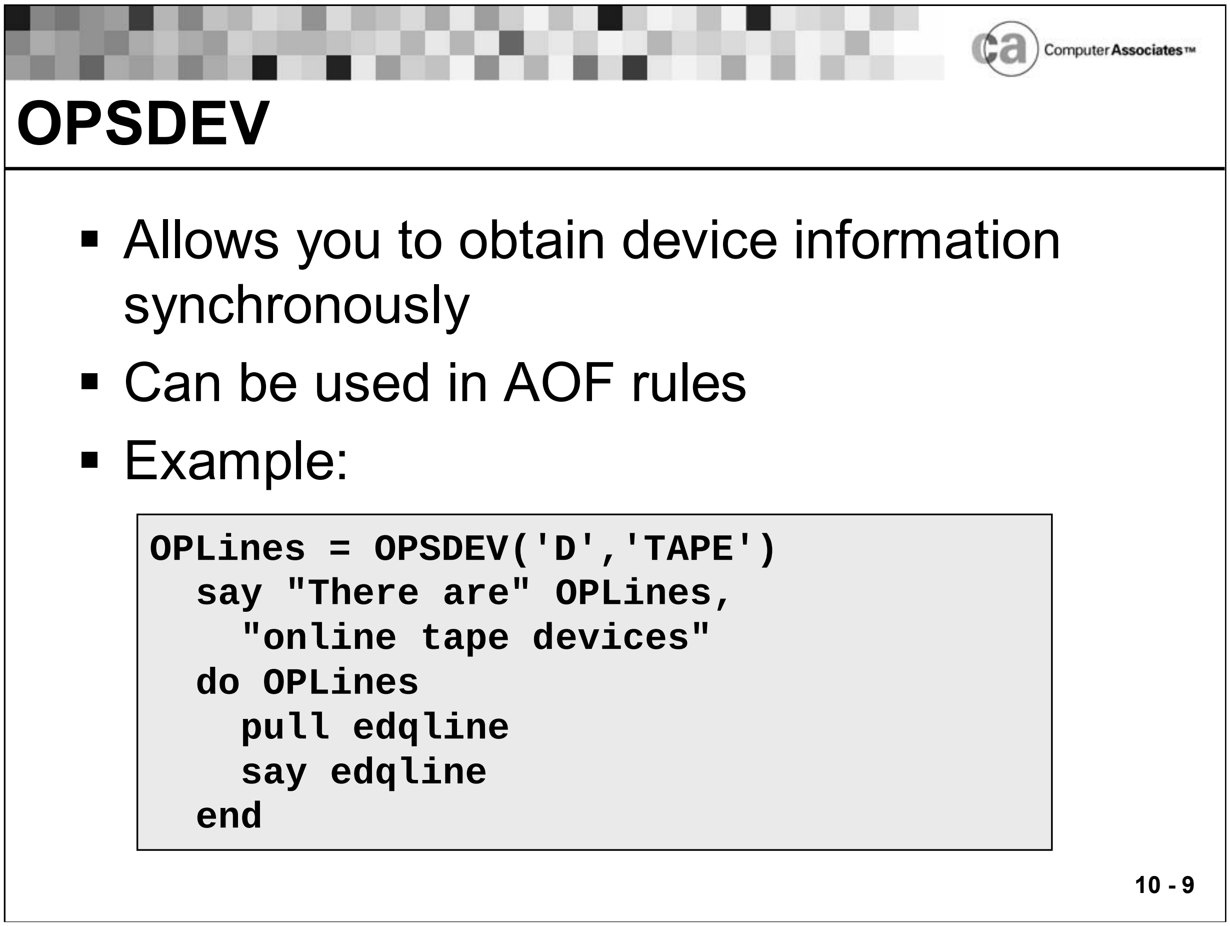
  Obtain information about z/OS system symbols.
- OPSWAIT

  Suspend the processing of an OPS/REXX program or a REQ rule for a specified period of time.
- OPSWORD

  Provide compatibility for translated CA-AutoMate/MVS rules that contain the &WORDnn and &MLWORDnn environmental variables.
- OPSWLM

  Retrieve information or set the state of a WLM scheduling environment.

Refer to your Unicenter CA-OPS/MVS documentation for specific information about OPS/REXX functions. This lesson covers the OPSDEV, OPSINFO, OPSTATUS, and OPSVALUE functions in detail.



OPSDEV

- Allows you to obtain device information synchronously
- Can be used in AOF rules
- Example:

```
OPLines = OPSDEV('D','TAPE')
  say "There are" OPLines,
    "online tape devices"
  do OPLines
    pull edqline
    say edqline
  end
```

10 - 9

OPSDEV

This function obtains device information synchronously. Therefore, you can use it in AOF rules. It has the following format:

```
var = OPSDEV(function,qualifier[,status])
```

The function argument describes the type of information retrieval that OPSDEV does. Possible values are:

- U - Extract device information by UCB. The qualifier is a four-character device number. You can use the % and/or * characters as wildcards. For example, the REXX statement below returns information about all online devices whose device numbers contain “123” in the first three positions:

```
Devices=OPSDEV('U','123*')
```

- V - Extract information by VOLSER. The qualifier is a six-character volume serial. You can use the % as a single character wildcard and the * symbol only as a suffix wildcard. For example, the REXX statement below returns information about all online DASD volumes whose volume serial begin with an S and whose third through fifth characters are RES:

  ```
  Devices=OPSDEV('V','S%RES*')
  ```

  Only DASD devices are searched for matching volume serials.
- D - Extract information about devices in general. The qualifier determines the type of device to return information about. The values can be:

  DASD

  TAPE

  UREC (unit record devices)

  COMM (communications devices)

  CTC (channel-to-channel adapters)

  TERM (terminals)

  (all devices)

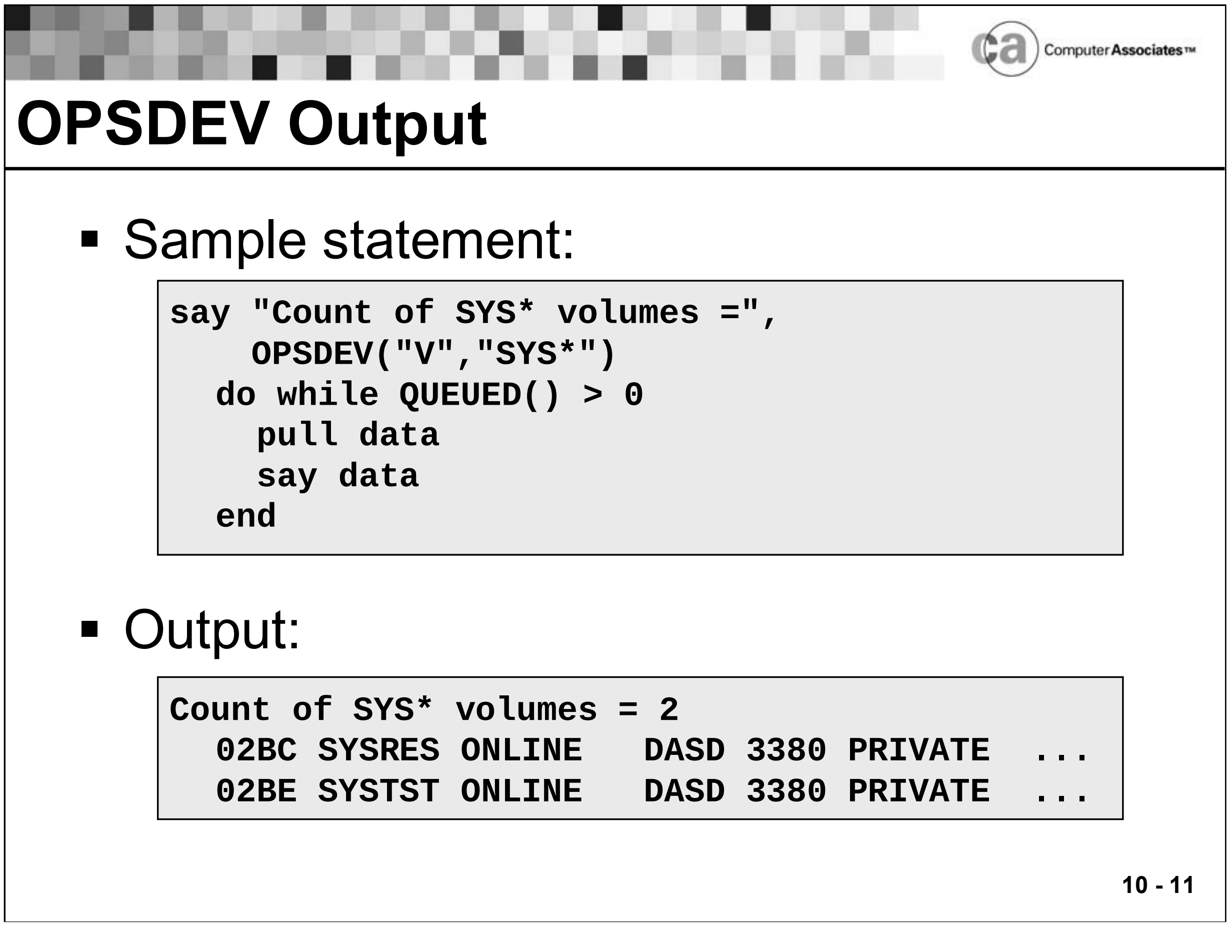
  For example, the REXX statement below returns information about all online channel-to-channel adapters:

  ```
  Devices=OPSDEV('D','CTC')
  ```

The qualifier specifies the devices for which information is returned.

The status value determines whether both online and offline or only online devices are to be considered. Possible values for status are:

- O - Only information on online devices is returned. (Default.)
- A - Information is returned on both online and offline devices.



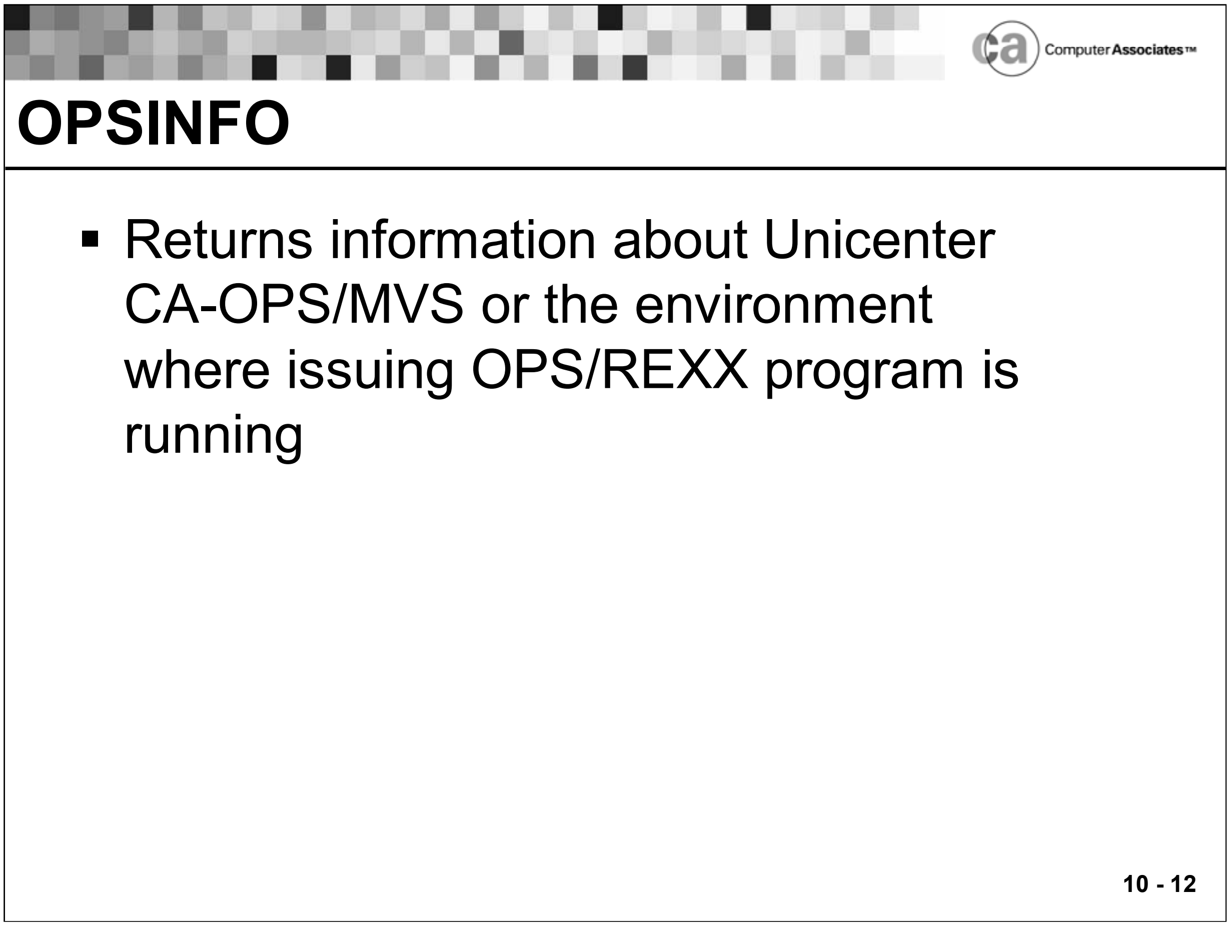
OPSDEV Output

- Sample statement:

```
say "Count of SYS* volumes =",
    OPSDEV("V","SYS*")
  do while QUEUED() > 0
    pull data
    say data
  end
```

- Output:

```
Count of SYS* volumes = 2
  02BC SYSRES ONLINE   DASD 3380 PRIVATE  ...
  02BE SYSTST ONLINE   DASD 3380 PRIVATE  ...
```



OPSINFO

- Returns information about Unicenter CA-OPS/MVS or the environment where issuing OPS/REXX program is running

10 - 12

OPSINFO

The OPSINFO function returns information about Unicenter CA-OPS/MVS and/or the environment where the issuing OPS/REXX program is running. It has the following format:

```
var = OPSINFO(string)
```

The following lists some of the more useful OPSINFO functions. For a complete list, see the documentation.

- var = OPSINFO('ACCOUNT')

  ACCOUNT information for the current address space. Multiple accounting fields are returned separated by blanks.

- var = OPSINFO('ASID')

  The z/OS address space identifier (ASID) number of the address space in which the Unicenter CA-OPS/MVS program issuing this function call is running. The value returned is two characters whose binary value is the ASID. If the OPS/REXX program is a message rule, OPS/REXX returns the ASID of the issuer of the WTO that triggered the rule.

- var = OPSINFO('CPUID')

  The six-character CPU identifier or the CPU where the OPS/REXX program or rule is running.

- var = OPSINFO('CPUTYPE')

  The four-character CPU type (for instance, 3090) of the CPU where the OPS/REXX program or rule is running.

- var = OPSINFO('CPUMODEL')

  The two-character CPU sub-model code of the CPU where the OPS/REXX program or rule is running. When running under VM this function returns the value FF.

- var = OPSINFO('DFPVERSION')

  The level of DFP running on the system where the OPSINFO function call was executed in the form version.release.modification level (for example, 3.2.0).

- var = OPSINFO('EXECPGM')

  The name of the program specified on the EXEC PGM=progname JCL statement for the current step.

- var = OPSINFO('EVENTTYPE')

  The type of event the rule is processing, such as MSG or DOM. This function is intended for use in external OPS/REXX functions that may be called by more than one kind of rule.

  When you use this function outside the rule environment (for example, in an OPS/REXX program in a server), it always returns the string “NONE.”

- var = OPSINFO('EXITTYPE')

  The name of the exit in which the AOF captured the event that triggered this rule. This function returns one of these values:

  ARM
  CA7
  CICS
  CNSV
  CPM
  DSN
  ERR
  IMS
  JES3
  MVS
  NIP
  NONE
  OMG
  TRAC

  A common use of OPSINFO('EXITTYPE') is to prevent processing a single message more than once in the IMS or JES3 environments.

- var = OPSINFO('IMSID')

  The IMS identifier (IMSID) associated with the address space where the OPS/REXX program is running. “NONE” is returned if the address space is not an IMS Control Region, a DLISAS Region, a DBRC address space, a Message Processing Region, or a Batch Message Processing Region.

  IMSID is of greatest interest to AOF message rules. For example, canceling an IMS BMP is dangerous because doing so can bring down all of IMS. A rule can use the OPSINFO function to:

  Determine whether the job issuing an z/OS message that usually calls for cancellation is an IMS-dependent region .

  Take some other action for these regions.

  Note: This function is available only if the IOF is licensed, installed, and active at your site. If the IOF is not installed or is inactive, this function always returns the string “NONE.”

- var = OPSINFO('IPLDATE')

  The date when z/OS was last IPLed (for example, 20030428 for April 28, 2003), in standard OPS/REXX date format S.

- var = OPSINFO('IPLDEVICE')

  The device number of the DASD volume from which the last IPL was performed (for example, 9C01). This device number can have four hexadecimal digits.

- var = OPSINFO('IPLTIME')

  The time when z/OS was last IPLed (for example, 13:30:10 for 1:30 p.m. plus 10 seconds), in standard OPS/REXX time format N.

- var = OPSINFO('IPLTYPE')

  A value representing how the system was last IPLed. This value is one of the following:

  CLPA - The system was IPLed using the CLPA option

  CVIO - The system was IPLed using the CVIO option

  WARM - The system was warm-started (IPLed without a complete shutdown)

- var = OPSINFO('IPLVOLSER')

  The volume serial of the DASD volume from which the last IPL was performed (for example, SYSRES).

- var = OPSINFO('JES')

  The name of the primary job entry subsystem. Typically, this value is either JES2 or JES3, but it can be any subsystem name that you have assigned to the primary JES (JES5, JES9, and so on). Contrast with the description of OPSINFO('JESTYPE').

- var = OPSINFO('JES3GBLNAME')

  One of these values:

  The JES3 global processor name if JES3 is the primary job entry subsystem.

  NULL if JES3 is not the primary job entry subsystem or JES3 is not active in the system.

- var = OPSINFO('JESTYPE')

  A value that indicates whether the primary JES subsystem is a JES2 or JES3 subsystem. The returned value is always either JES2 or JES3, regardless of the actual subsystem name. For example, if the primary JES is called JES9 but it is a JES2 subsystem, the following values are returned:

  ```
  OPSINFO(‘JES’)     = ‘JES9’
  OPSINFO(‘JESTYPE’) = ‘JES2’
  ```

  Note: The value returned by OPSINFO(‘JESTYPE’) may not be valid when the primary JES is not active in the system (for example, immediately following an IPL).

- var = OPSINFO('JOBNAME')

  The JOBNAME associated with the address space where the OPS/REXX program is running. For a message rule being processed in the z/OS SS09 exit, OPS/REXX returns the job name of the issuer of the WTO that triggered the rule. This means that for a reissued message that originated on another system the job name is the one that actually issued the WTO on the originating system.

- var = OPSINFO(‘LOCMSTCONSNM’)

  The first console name that has master authority on the local system. If there is no console that has master authority on the local system, the function returns the name of the sysplex master console. When a “no consoles” condition exists in the system, this function returns a null string.

  The OPSINFO(‘LOCMSTCONSNM’) function applies only to systems running MVS/ESA 4.1.0 or higher.

- var = OPSINFO('LPARNAME')

  The LPAR name of the processor configuration. Returns a null string if not in LPAR mode or if used on any OS/390 version below 5.2.2.

- var = OPSINFO('MAINPRM')

  The character string that was passed to Unicenter CA-OPS/MVS from the PARM field on its EXEC card when it was started. This string can contain up to 100 characters.

- var = OPSINFO('MODULE')

  The name of the module that triggered the AOF event. For example, in a message rule it returns the name of the module (program) that issued the WTO. When used in an OPS/REXX program running in a TSO address space or in a server it returns the name of an Unicenter CA-OPS/MVS module.

- var = OPSINFO('MSFID')

  The local MSF ID of the current copy of Unicenter CA-OPS/MVS. If the MSF is not installed or has not been started, a blank value is returned.

- var = OPSINFO('MSTCONSNM')

  The name of the current MCS master console. This data is available only if you have MVS/SP 4.1.0 or a higher level of z/OS on your system. When invoked under older releases of MVS, this function returns a null value. When a “no consoles” condition exists in the system, this function returns a null string.

- var = OPSINFO('MVSVERSION')

  The level of z/OS on which the OPSINFO function call was executed in the form version.release.modification level.

- var = OPSINFO('PRODUCTSTARTS')

  The number of times that the current Unicenter CA-OPS/MVS subsystem has been started since the last IPL of z/OS, or 0 if the subsystem has not been started since the last IPL.

  When used in an OPS/REXX program, this function returns a valid value even if Unicenter CA-OPS/MVS subsystem is inactive.

  Note: The only time that you can absolutely infer that Unicenter CA-OPS/MVS is down from this function is if it returns 0. If it returns any other value, it could be either up or down. To determine whether a particular product subsystem is active, check if the RC value set by this command is 0:

  ```
  ADDRESS AOF "SUBSYS ssid"
  ```

- var = OPSINFO('PRODUCTSTATUS')

  One of the following values:

  INIT - Unicenter CA-OPS/MVS is initializing.

  ACTIVE - Unicenter CA-OPS/MVS is active and processing system events.

  TERM - Unicenter CA-OPS/MVS is terminating .

  Only rules can invoke this function. It returns a null string when a REXX program tries to invoke it. Typically, you might use this function:

  In the initialization section of a rule, to determine whether this is Unicenter CA-OPS/MVS initial start-up.

  In the termination section of a rule, to determine whether Unicenter CA-OPS/MVS is terminating.

- var = OPSINFO('PROGRAM')

  In any environment other than a rule running under the AOF, OPSINFO('PROGRAM') returns the name of the member from which OPS/REXX read the REXX program. (If the OPS/REXX program was run from a sequential data set, which is rare, then OPSINFO('PROGRAM') returns a null string.) In the AOF environment, OPSINFO('PROGRAM') returns ruleset.rule.

- var = OPSINFO('PROGRAMMER')

  The programmer information associated with the current address space. In the AOF environment (except when running in a server) the information returned is for the address space that triggered the AOF event.

- var = OPSINFO('SMFID')

  The SMF identifier that your site has assigned to the z/OS image on which Unicenter CA-OPS/MVS is running. z/OS gets this value at IPL time from SYS1.PARMLIB(SMFPRMnn).

- var = OPSINFO('STATEMANSTATUS')

  The current status of the System State Manager task. Possible values are:

  ACTIVE - System State Manager is active.

  INACTIVE - System State Manager is not active.

  NONE - System State Manager is not being used.

  PASSIVE - System State Manager is active but is not monitoring resources.

  NOT-AVAILABLE - The OPS/MVS product is not active.

- var = OPSINFO('STEPNAME')

  The step name or alternate task ID of the current address space. For example, if you issue the command S CICS.CICSA, the step name for that address space is CICSA.

- var = OPSINFO('SUBSYS')

  The four-character z/OS subsystem name that Unicenter CA-OPS/MVS is using. This value from the first four characters of the PARM field on the EXEC statement in the JCL procedure used for Unicenter CA-OPS/MVS. Default subsystem name is OPSS.

- var = OPSINFO('SYSPLEX')

  The z/OS sysplex name from SYS1.PARMLIB(COUPLExx) or SYS1.PARMLIB(LOADxx). Returns a null string if used on any OS/390 version below 4.1.0.

- var = OPSINFO('TSOEVERSION')

  The TSO/E release level in the form v.rr.m (for example, 2.05.0). If TSO/E is not installed, this function returns the value N/A.

- var = OPSINFO('USSPID')

  The UNIX System Services process ID number (PID), in decimal format, for the current z/OS task. The primary use of this function it to capture the PID from long-running USS processes that issue messages resulting in AOF message events. Possible values are:

  Nnnnnnnnnn - Process ID number of current z/OS task.

  Important! This number is not the same as the PID of the parent process when UNIX services fork or spawn are used to issue a message. These UNIX services may generate a short-term process resulting in a message that triggers an AOF message event.

  N/A - No process ID exists for the current task.

  Note: To return the process ID number, you must have at least OS/390 version 1, release 2 or higher.

- var = OPSINFO('VERSION')

  An eight-byte string that provides the version, release, and modification level of Unicenter CA-OPS/MVS. The format of this string is vv.rr.mm where vv is the version number, rr is the major release number, and mm is the modification level number within that release number. Each time CA makes a new “level set” maintenance tape available, the modification level within the product version string changes. For example, the third maintenance tape that CA issues for Unicenter CA-OPS/MVS 04.02.00 might have a version string of “04.02.03.”

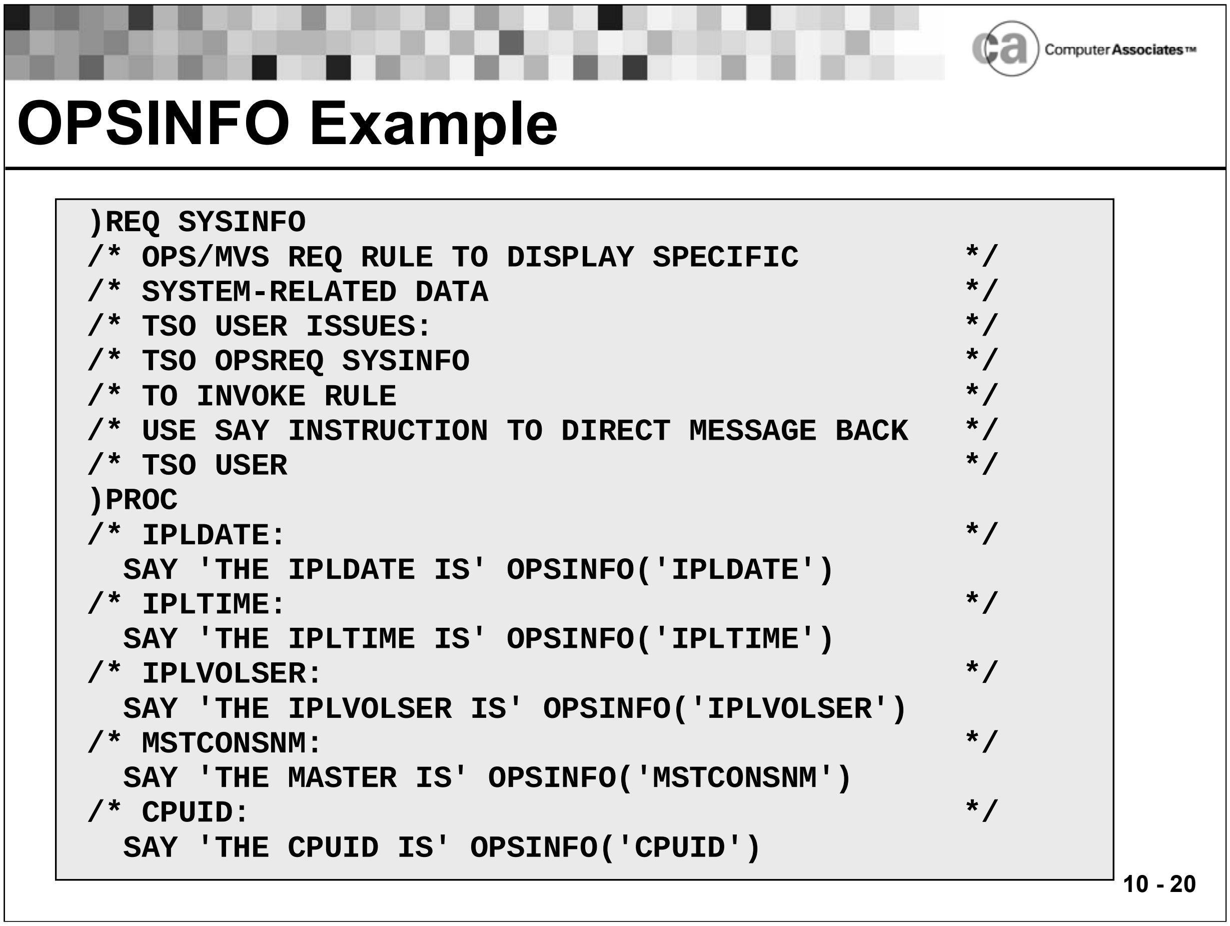
Thus, when using the OPSINFO('VERSION') function to differentiate between code that may run on multiple versions of Unicenter CA-OPS/MVS, take into account that the modification level changes with each maintenance tape.

For example, to check for code that can run only with version 3.2, use this function:

```
if SUBSTR(OPSINFO("VERSION"),1,6)=="03.02."
then ...
```

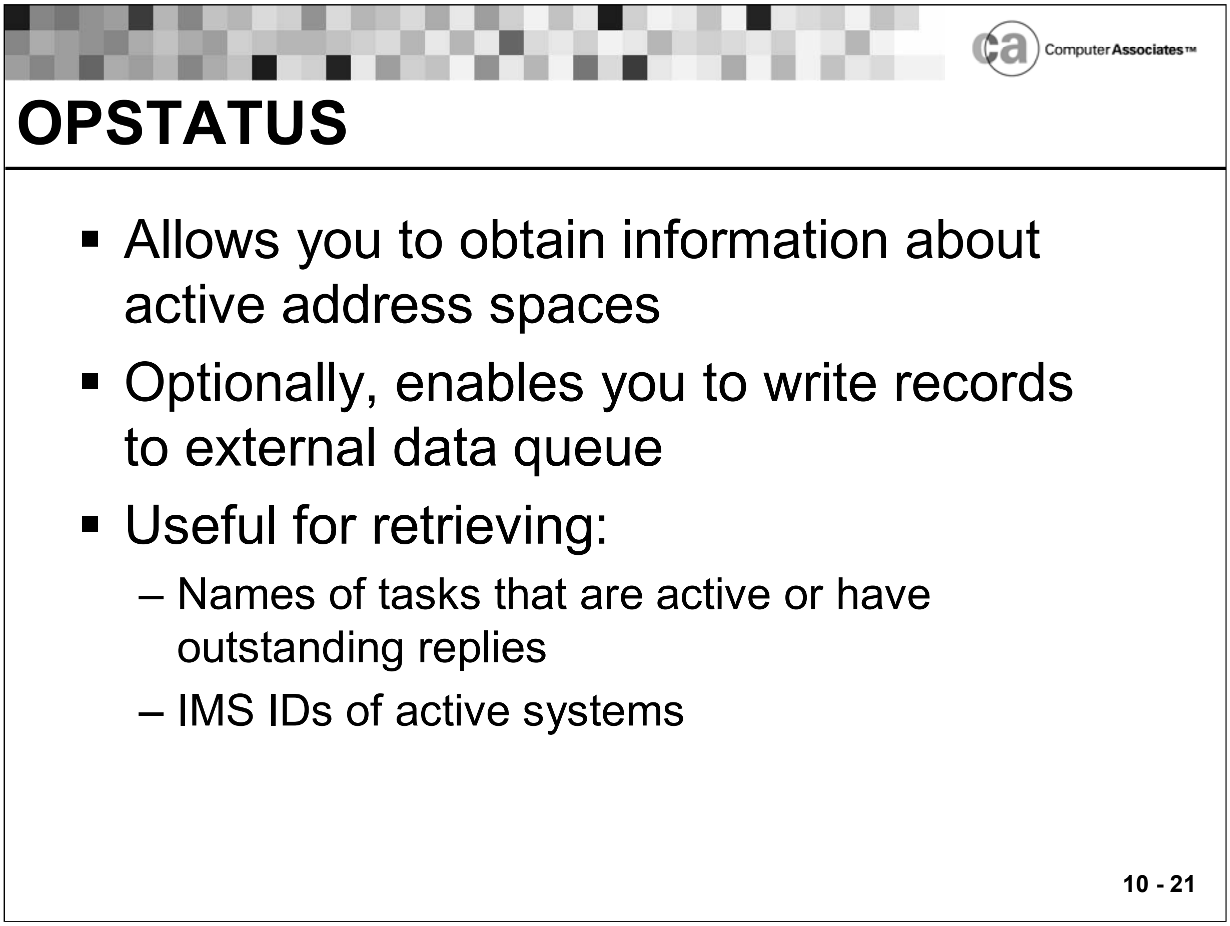
To check for code that can run with version 3.2 or higher levels, use this function:

```
if SUBSTR(OPSINFO("VERSION"),1,6)>="03.02."
then ...
```



OPSINFO Example

```
)REQ SYSINFO
/* OPS/MVS REQ RULE TO DISPLAY SPECIFIC          */
/* SYSTEM-RELATED DATA                           */
/* TSO USER ISSUES:                              */
/* TSO OPSREQ SYSINFO                            */
/* TO INVOKE RULE                                */
/* USE SAY INSTRUCTION TO DIRECT MESSAGE BACK    */
/* TSO USER                                      */
)PROC
/* IPLDATE:                                      */
  SAY 'THE IPLDATE IS' OPSINFO('IPLDATE')
/* IPLTIME:                                      */
  SAY 'THE IPLTIME IS' OPSINFO('IPLTIME')
/* IPLVOLSER:                                    */
  SAY 'THE IPLVOLSER IS' OPSINFO('IPLVOLSER')
/* MSTCONSNM:                                    */
  SAY 'THE MASTER IS' OPSINFO('MSTCONSNM')
/* CPUID:                                        */
  SAY 'THE CPUID IS' OPSINFO('CPUID')
```



OPSTATUS

- Allows you to obtain information about active address spaces
- Optionally, enables you to write records to external data queue
- Useful for retrieving:
  - Names of tasks that are active or have outstanding replies
  - IMS IDs of active systems

10 - 21

OPSTATUS

The OPSTATUS function returns a count of the address spaces currently in the system or in the “*LOGON*” state, WTORs, or IMS systems that are currently in the system and that match the selection criteria. OPSTATUS also optionally writes one or more records to the external data queue.

OPSTATUS is especially useful for retrieving either the names of tasks that are active or have outstanding replies, or the IMS IDs of active IMS systems.

Note: Using the OPSTATUS function to count the number of batch jobs from a $HASP message rule may not provide you with accurate results. The message may be issued on behalf of a batch job before the OPSTATUS function counts it or after OPSTATUS counts it. Batch jobs may show up as active before the program is executing, or after the program has finished executing but before JES has finished handling work on its behalf.

The OPSTATUS function has this format:

```
var = OPSTATUS( func, subfunc, entity)
```

The func argument specifies the type of entity whose status is to be queried; it has one of these values:

- A - Address spaces.
- I - IMS systems.
- J - Batch jobs.
- P - Attached APPC transactions (ATX).
- R - Outstanding replies (WTORs).
- S - Started tasks and system address spaces.
- T - TSO users.

The subfunc argument specifies one of these information types:

- A - Return the count only.
- L - Return the count and add a WTOR or IMS record to the external data queue for each WTOR or IMS system found. A subfunc of L is meaningful only for func values of R (WTORs) and I (IMS systems).
- I - Return the count and do the following:

  For func values of A, J, P, S, or T, add an ASID record to the external data queue for each address space found.

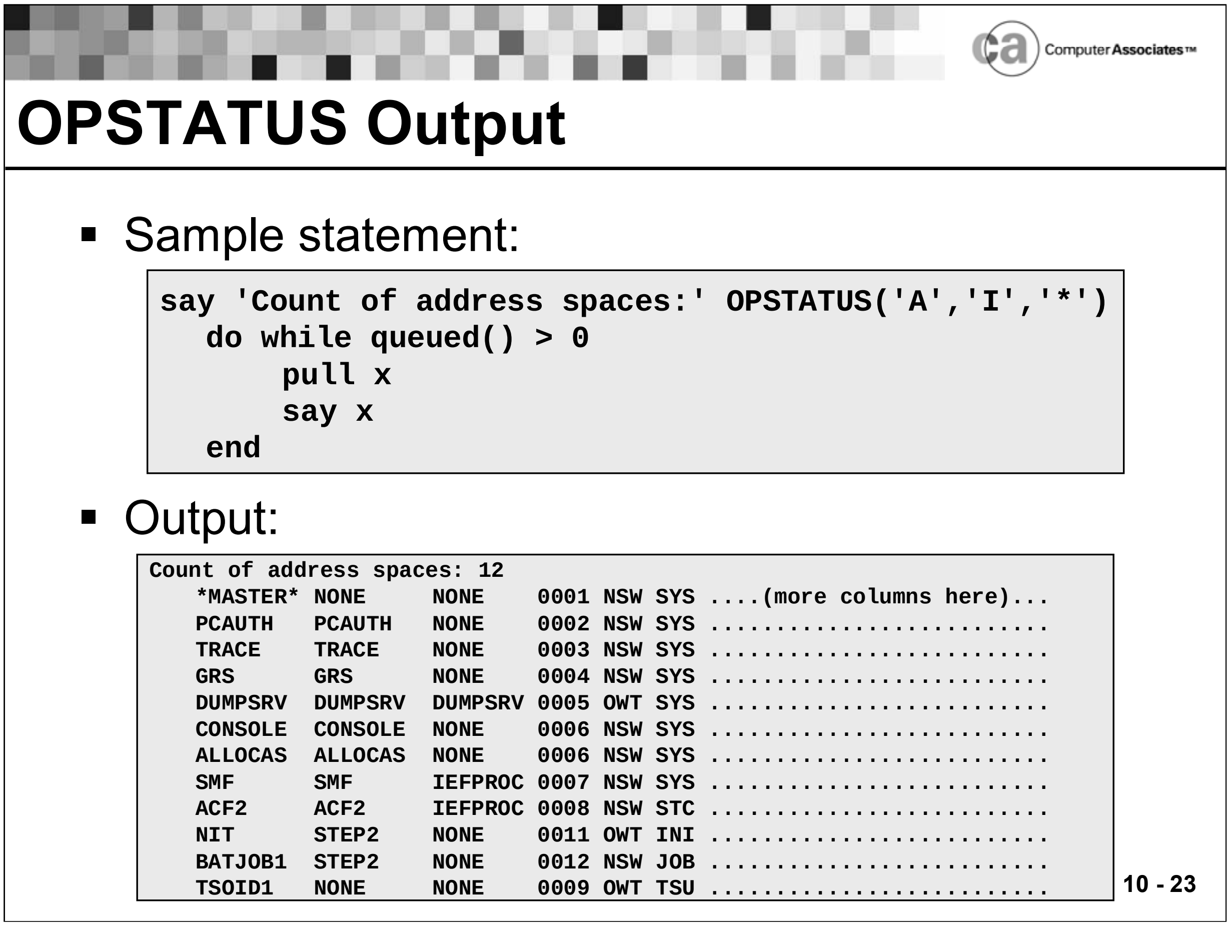
  For a func value of I, add an IMS record (followed by an ASID record) to the external data queue for each IMS system found. The ASID record identifies the IMS system.

  For the func value R, add a WTOR record (followed by an ASID record) to the external data queue for each WTOR found. The ASID record identifies the issuer of the WTOR.
- W - Returns an external data queue record for each ASID, JOB, STC, or TSO address that matches the selection criteria.

The entity argument restricts the returned information to only those entities with names matching the value of entity. Entity can be:

- An asterisk to indicate all entities of a given type. (Be sure to use this when searching for IDs currently in the *LOGON* state.)
- A string of characters ending with an asterisk to return information about only the entities with names that begin with the specified string.
- A string of characters with no asterisk to return information about only entities with names that exactly match the string.

For all function codes except I, the entity name refers to the jobname/TSO ID/taskname of each address space. For function code I, entity names refer to the four-character IMS ID of each IMS system.



OPSTATUS Output

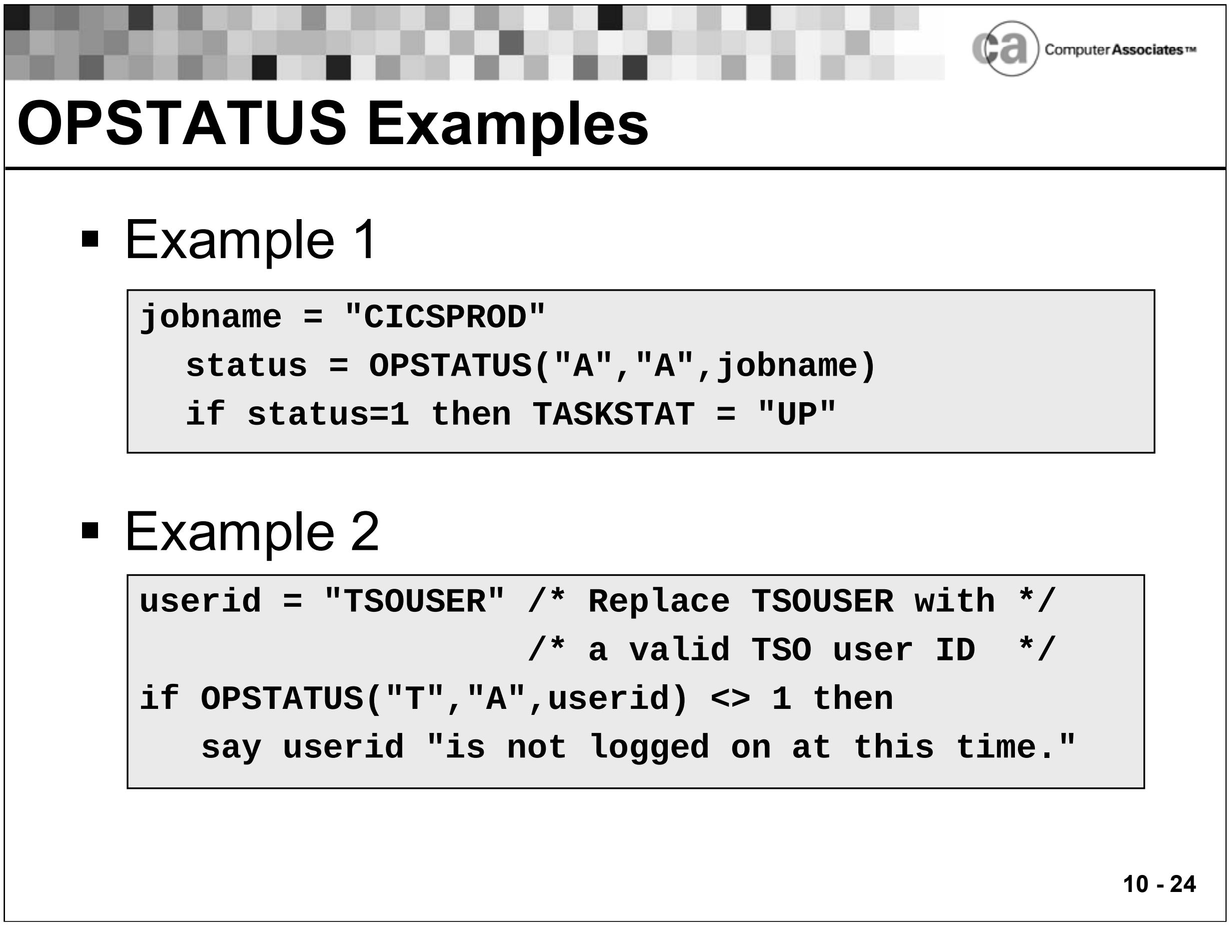
- Sample statement:

```
say 'Count of address spaces:' OPSTATUS('A','I','*')
  do while queued() > 0
      pull x
      say x
  end
```

- Output:

```
Count of address spaces: 12
   *MASTER* NONE     NONE    0001 NSW SYS ....(more columns here)...
   PCAUTH   PCAUTH   NONE    0002 NSW SYS ..........................
   TRACE    TRACE    NONE    0003 NSW SYS ..........................
   GRS      GRS      NONE    0004 NSW SYS ..........................
   DUMPSRV  DUMPSRV  DUMPSRV 0005 OWT SYS ..........................
   CONSOLE  CONSOLE  NONE    0006 NSW SYS ..........................
   ALLOCAS  ALLOCAS  NONE    0006 NSW SYS ..........................
   SMF      SMF      IEFPROC 0007 NSW SYS ..........................
   ACF2     ACF2     IEFPROC 0008 NSW STC ..........................
   NIT      STEP2    NONE    0011 OWT INI ..........................
   BATJOB1  STEP2    NONE    0012 NSW JOB ..........................
   TSOID1   NONE     NONE    0009 OWT TSU ..........................
```

10 - 23



OPSTATUS Examples

- Example 1

```
jobname = "CICSPROD"
  status = OPSTATUS("A","A",jobname)
  if status=1 then TASKSTAT = "UP"
```

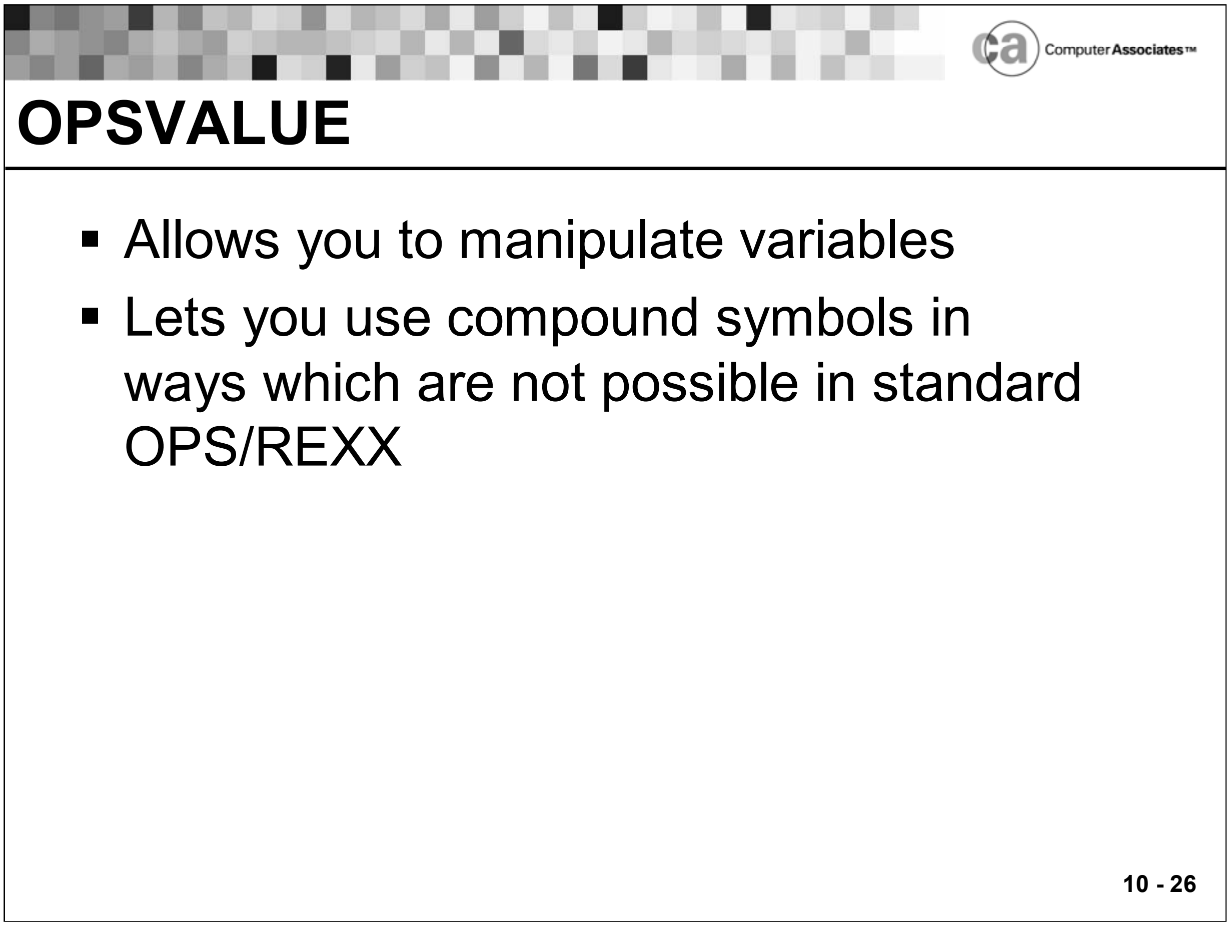
- Example 2

```
userid = "TSOUSER" /* Replace TSOUSER with */
                   /* a valid TSO user ID  */
if OPSTATUS("T","A",userid) <> 1 then
   say userid "is not logged on at this time."
```

OPSTATUS Examples (continued)

- Example 3

```
GETWTOR=OPSTATUS('R','L','QAH02HXT')
  PULL RECORD
  REPLYID = WORD(RECORD,1)
  ADDRESS OPER
    "COMMAND(R "REPLYID",SHUTDOWN) NOOUTPUT"
```



OPSVALUE

- Allows you to manipulate variables
- Lets you use compound symbols in ways which are not possible in standard OPS/REXX

10 - 26

OPSVALUE

Using the OPSVALUE function, you can manipulate compound symbols in ways which are not possible in standard OPS/REXX. For example, the OPSVALUE function lets you use compound symbols, especially global compound symbols, as a kind of database. It has the following format:

```
var = OPSVALUE(derivedname[,[actioncode][,[newvalue][,oldvalue]]])
```

Limits for Global Variable Stems

- You should not create too many global variables under a single global variable stem. If you do, you will no longer be able to view the global variables under OPSVIEW option 4.8 or access them using the OPSVALUE function.
- The absolute product limit is 32,768 variables under a single global variable stem. However, in practice, CA strongly recommends that no more than 10,000 global variables exist at any given instant under a single global variable stem.
- Note: There is a restriction in viewing too many variables when going cross-system. Trying to access several hundred variables may cause a time-out on the MSF link.

OPSVALUE Arguments

- The argument derivedname gives the name of the symbol to be acted on. When you use this argument without quotation marks, simple symbols (which are case-sensitive) following the stem are replaced by their values. Although the derived name may be as long as 84 bytes, global variable rules can process only the first 50 bytes of the name. Therefore, you may want to limit the length of global variable names to 50 bytes.
- The actioncode specifies the action to be taken on that symbol. The newvalue argument supplies the new value (if any) to assign to the symbol, and oldvalue fetches the value of the symbol before the actioncode action takes place.

OPSVALUE and Cross-system Operations

- To use the OPSVALUE function for cross-system operations, change the default system name by using the following command:

```
ADDRESS OPSCTL "MSF DEFAULT SYSTEM(sysname)"
```

- Any subsequent OPSVALUE function is routed to the sysname system. You must specify an individual system name on the ADDRESS OPSCTL MSF DEFAULT host command that is associated with the OPSVALUE function. Values of ALL and EXT are invalid when used with OPSVALUE.
- To return to the local system, use this command:

```
ADDRESS OPSCTL "MSF DEFAULT SYSTEM(*)"
```

- For more information about the ADDRESS OPSCTL MSF DEFAULT command, see your Unicenter CA-OPS/MVS documentation.

Actions OPSVALUE Takes

- OPSVALUE returns a value from the function call, and, in the case of some action codes, also places information in the external data queue.
- You can specify the actioncode values shown below. If you omit the actioncode, OPS/REXX uses the code V by default.

6 (Delete single variable)

Removes the node specified by the derived name without removing any of its subnodes.

Returns 1 if the node was deleted; returns 0 if the node was not found.

Does not change the external data queue.

Does not allow other accessors of compound symbols to see partially updated symbol names.

```
RTVL = OPSVALUE(derivedname,'6')
```

A (Add)

Adds a number specified by increment to the existing compound symbol given by derivedname.

Returns the sum of the compound symbol and the increment.

Does not change the external data queue.

All references to the compound symbol are serialized during the ADD operation. That is, you can use this function safely to increment a counter that is set by concurrent tasks.

```
RTVL = OPSVALUE(derivedname,'A',increment)
```

C (Compare and update)

Updates a compound symbol after verifying its current value.

Safely updates global compound symbols shared by more than one rule or global compound symbols that multiple copies of the same rule might access and update.

Does not change the OPS/REXX external data queue.

Returns the REXX “true” value (1), if the comparison found the symbol’s pre-action value to be equal to old value and the compound symbol was updated, or the REXX “false” value (0), if the comparison found unequal values and therefore did not update the value of the compound symbol.

Serializes the compare and update operations for global variables.

Example: To perform the Compare and Update action, four rather than three operands must be used with the 'C' code. The syntax is as follows:

```
RTVL = OPSVALUE(derivedname,'C',newvalue,oldvalue)
```

Usage Example—The following example demonstrates how a counter could be increased incrementally using the Compare and Update action code.

```
)MSG IEF13241
)INIT
  GLOBAL.IEF13241 = 0
)PROC
  DENA = 'GLOBAL.IEF13241‘
  DO WHILE ¬ OPSVALUE(DENA,'C',GLOBAL.IEF13241+1,GLOBAL.IEF13241)
  END
```

D (Drop)

Performs the OPS/REXX DROP operation on the compound symbol specified by derivedname. The compound symbol is reset to its “uninitialized” value; that is, its derived name. If derivedname is the name of a stem, then all compound symbols belonging to that stem are not just dropped, but also rendered “nonexistent” and the virtual storage allocated to them is released.

Returns the value of derivedname.

Does not change the external data queue.

All other references either see the compound symbol as it existed before the DROP operation began, or as it is after the DROP operation completes.

```
RTVL = OPSVALUE(derivedname,'D')
```

E (Existence)

Checks to see whether a given global variable exists.

Does not change the OPS/REXX external data queue.

Returns status of a given global variable as:

I - Initialized

U - Uninitialized

N - does Not exist

```
RTVL = OPSVALUE('derivedname','E')
```

Note: For most types of variables, N and U have interchangeable meanings. However, for global variables, N means that no storage exists for a variable; and U means that the variable exists in storage, but is uninitialized and is set to the value of its name.

F (Find)

Checks to see whether a given global variable exists. The F action is more efficient and more reliable than using the E and O functions together.

Returns the status of a given global variable as one of these characters:

I - Initialized

U - Uninitialized

N - does Not exist

When the returned value is not N (meaning that the derived name exists), the value of the node is returned on the external data queue. The maximum length of a string pulled from the external data queue is 350 bytes. OPS/MVS truncates longer values.

```
RTVL = OPSVALUE(derivedname,'F')
```

H (High level security)

Provides more efficient access to global variables for an OPS/REXX program by establishing an authorization status for subsequent global variable access requests that the program makes. (Although you can use this function from any environment that supports OPSVALUE—such as REXX programs, rules, GEM, and OPSLINK—it does not perform useful work unless you use it in an OPS/REXX program.)

The value you specify for derivedname must be either GLOBAL.READONLY or GLOBAL.READWRITE. No other values are permitted.

Returns a value describing the authorization status of the OPS/REXX program:

AUTH - The request is permitted.

NOTAUTH - The request is denied (no error message is issued).

When the AUTH value is returned, no subsequent OPSVALUE calls from the OPS/REXX program (and any lower level subroutines it calls) create OPSGLOBAL security events of the type corresponding to the authority level obtained on the preceding high level security call. In other words:

If you specify GLOBAL.READWRITE on the function, and the returned value is AUTH, no subsequent OPSVALUE calls create security events.

If you specify GLOBAL.READONLY on the function, and the returned value is AUTH, no subsequent OPSVALUE access-type calls create security events.

This function results in an OPSGLOBAL product security event (SEC rule). The SEC.AUGLOPCH is set to H, and the SEC.AUGLRQTY variable is set to:

A (Access) - If you specify GLOBAL.READONLY

U (Update) - If you specify GLOBAL.READWRITE

```
RTVL = OPSVALUE('GLOBAL.READONLY','H')
RTVL = OPSVALUE('GLOBAL.READWRITE','H')
```

I (Information)

Returns to the external data queue information about all of the immediate subnodes of the derivedname.

The derivedname value must be a compound symbol node. The return value is the number of immediate subnodes that exist. The external data queue contains two lines per subnode: the first line contains the next segment of the derived name, and the second line contains statistics about the derived name. The second line returned for each derived name contains the information shown below. (The first piece of information indicates the word number, the second indicates the length of the word, and the third describes the word.)

– 1, 8 - The number of subnodes under this subnode.

– 2, 10 - Creation date (in the form yyyy/mm/dd).

– 3, 8 - Creation time (in the form hh:mm:ss).

– 4, 17 - Creation ruleset.rule or program name.

– 5, 8 - Creation Jobname/Taskname/TSO ID.

– 6, 10 - Last modification date (in the form yyyy/mm/dd).

– 7, 8 - Last modification time (in the form hh:mm:ss).

– 8, 17 - Last modification ruleset.rule or program name.

– 9, 8 - Last modification Jobname/Taskname/TSO ID.

– 10, 1 - This word, which is reserved, always contains the value 0; provides compatibility with programs expecting a numeric value.

– 11, 8 - Number of updates to this node.

– 12,10 - Last access date (in the form yyyy/mm/dd) or the string NONE if the variable was created under an older version of the product; permits users to determine when global variables have not been used in a long time and thus may be eligible for deletion.

Returns the number of subnodes listed in the external data queue.

Places two lines per subnode in the external data queue.

Returns no partially updated symbol names.

```
RTVL = OPSVALUE(derivedname,'I')
```

K (subtree count)

Returns a count of all the subnodes of the derivedname.

The result value returned by this action is the same value returned by the “S” or “T” action code. However, the external data queue is not modified.

```
RTVL = OPSVALUE(derivedname,'K')
```

L (List)

Lists the derived names of all the immediate subnodes of derivedname by placing them on the external data queue.

The results of this action illustrate the difference between dropped symbols (processed by action D) and removed symbols (processed by action R). Dropped symbols still exist, so the List action can find them. The List action does not return removed symbols.

Returns the number of subnodes listed in the external data queue.

Places a list of subnodes of the specified nodes in the external data queue.

Returns no partially updated symbol names.

```
RTVL = OPSVALUE(derivedname,'L')
```

O (Obtain)

Obtains the value of a global variable. If the global variable does not exist, OPS/REXX returns an error.

Does not change the external data queue.

```
RTVL = OPSVALUE(derivedname,'O')
```

R (Remove)

Removes the node specified by derivedname and all of its subnodes. Once a node is removed, it ceases to exist.

Returns the number of subnodes removed.

Does not change the external data queue.

Does not allow other accessors of compound symbols to see partially updated symbols.

```
RTVL = OPSVALUE(derivedname,'R')
```

S (Subtree)

Lists the derived names of all the subnodes of derivedname in the external data queue. Action code S is similar to code L with two differences:

– OPS/REXX places the entire global variable name in the external data queue.

– All subnodes of the derived name are listed.

Returns the number of subnodes listed in the external data queue.

Places the entire global variable name in the external data queue.

Returns no partially updated symbol names.

```
RTVL = OPSVALUE(derivedname,'S')
```

T (subTree/info)

Returns to the external data queue information on all the subnodes of the derivedname.

The derivedname value parameter must be a compound symbol node. The return value is the number of subnodes that exist. The external data queue contains two lines per subnode: the first line contains the next segment of the derived name, and the second line contains statistics about the derived name. (The second line contains the information listed blow—the first piece of information indicates the word number, the second indicates the length of the word, and the third describes the word.)

– 1, 8 - The number of subnodes under this subnode (always contains a zero).

– 2, 10 - Creation date (in the form yyyy/mm/dd).

– 3, 8 - Creation time (in the form hh:mm:ss).

– 4, 17 - Creation ruleset.rule or program name.

– 5, 8 - Creation Jobname/Taskname/TSO ID.

– 6, 10 - Last modification date (in the form yyyy/mm/dd).

– 7, 8 - Last modification time (in the form hh:mm:ss).

– 8, 17 - Last modification ruleset.rule or program name.

– 9, 8 - Last modification Jobname/Taskname/TSO ID.

– 10, 8 - This word, which is reserved, always contains the value 0; provides compatibility with programs expecting a numeric value.

– 11, 8 - Number of updates to this node.

– 12,10 - Last access date (in the form yyyy/mm/dd) or the string NONE if the variable was created under an older version of the product; permits users to determine when global variables have not been used in a long time and thus may be eligible for deletion.

Action code T resembles code I with three differences:

- The entire global variable name goes into the external data queue.

– All subnodes of the derived name are listed.

– The “Number of Subnodes” field on the second line of the pair of messages in the external data queue for each node always contains zero.

Returns the number of subnodes listed in the external data queue.

Places in the external data queue two lines per subnode and the entire global variable name.

Returns no partially updated symbol names.

```
RTVL = OPSVALUE(derivedname,'T')
```

U (Update)

Assigns newvalue as the value of the compound symbol specified by derivedname. If the compound does not exist, OPS/REXX creates it then gives it the new value.

Returns the variable specified by newvalue.

Does not change the external data queue.

Prevents others accessing compound symbols from seeing partially updated symbols.

```
RTVL = OPSVALUE(derivedname,'U',newvalue)
```

V (Value)

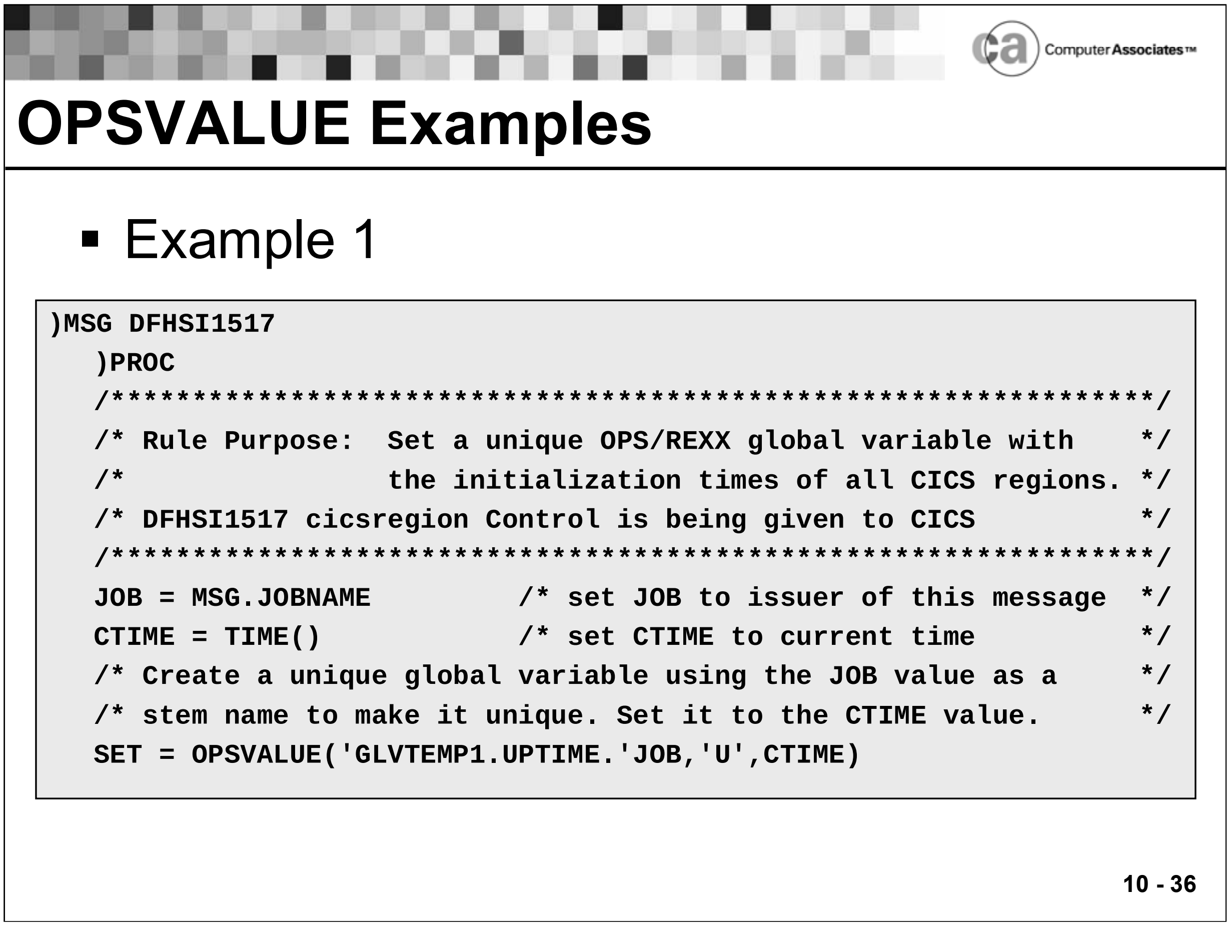
Returns the current value of the node specified by derivedname. If the node does not exist, OPS/REXX creates it but assigns it no value (giving the symbol the same value as its name).

Returns the value of the specified compound symbol.

Does not change the external data queue.

Prevents the issuer of OPSVALUE from seeing partially updated symbols.

```
RTVL = OPSVALUE(derivedname,'V')
```

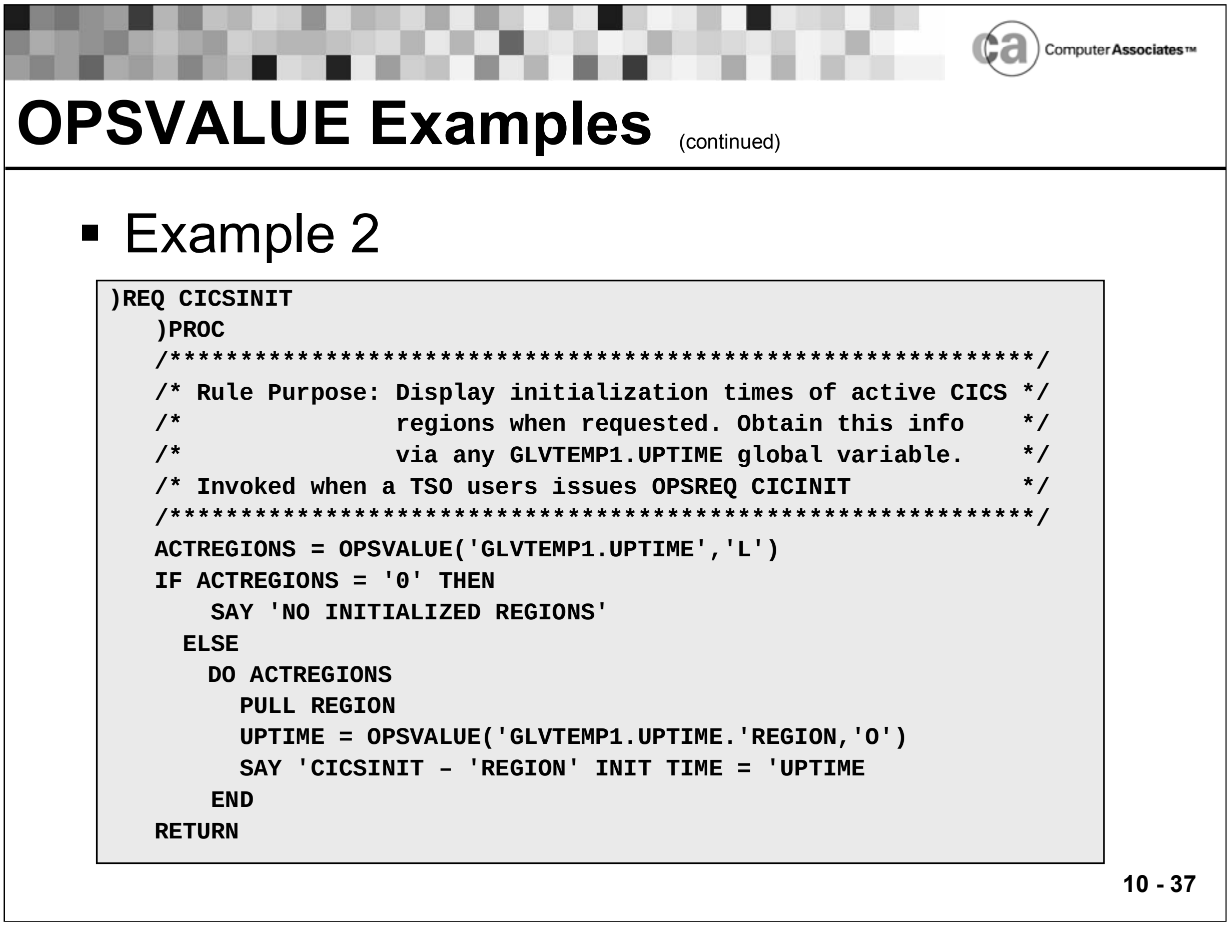


OPSVALUE Examples

- Example 1

```
)MSG DFHSI1517
   )PROC
   /****************************************************************/
   /* Rule Purpose:  Set a unique OPS/REXX global variable with    */
   /*                the initialization times of all CICS regions. */
   /* DFHSI1517 cicsregion Control is being given to CICS          */
   /****************************************************************/
   JOB = MSG.JOBNAME         /* set JOB to issuer of this message  */
   CTIME = TIME()            /* set CTIME to current time          */
   /* Create a unique global variable using the JOB value as a     */
   /* stem name to make it unique. Set it to the CTIME value.      */
   SET = OPSVALUE('GLVTEMP1.UPTIME.'JOB,'U',CTIME)
```

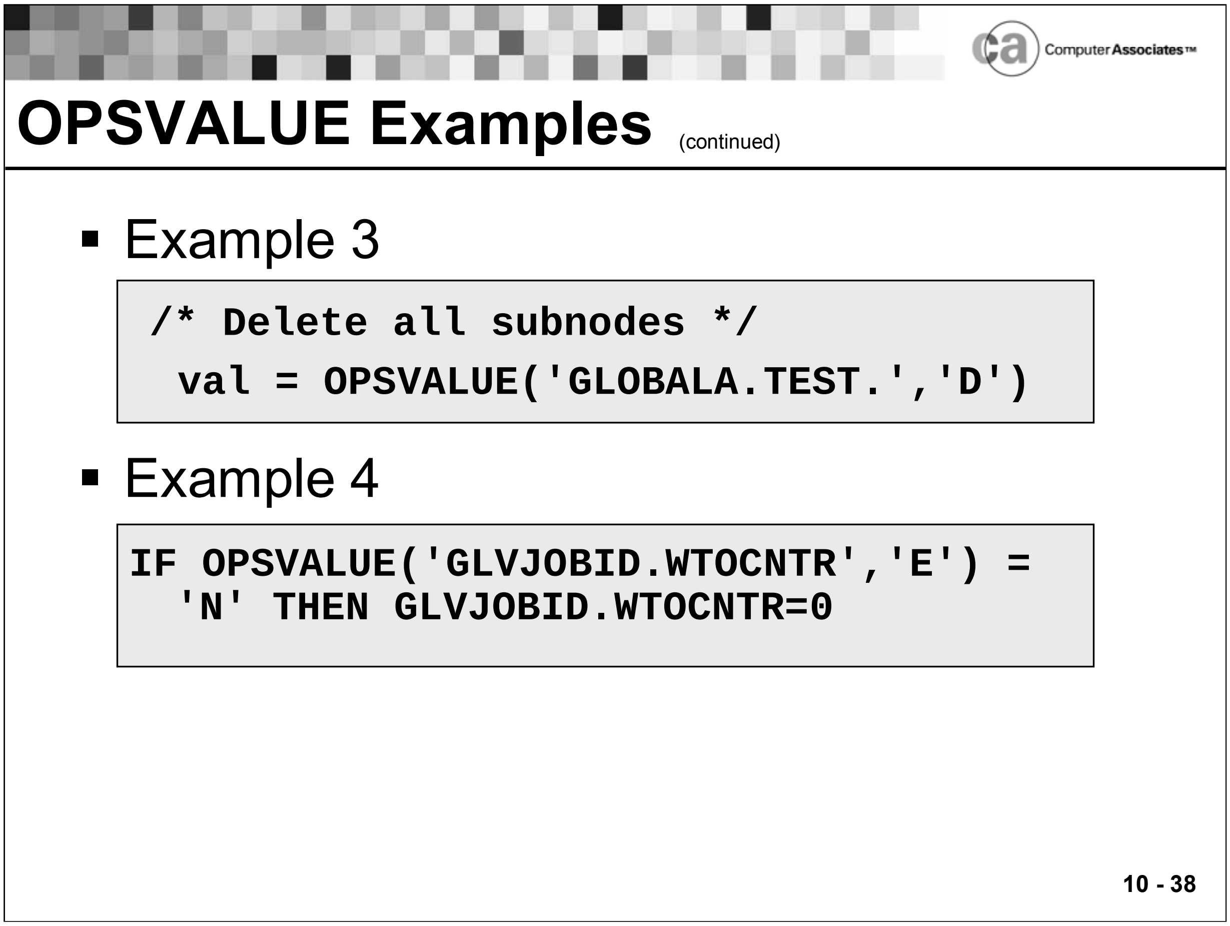
10 - 36



OPSVALUE Examples (continued)

- Example 2

```
)REQ CICSINIT
   )PROC
   /*************************************************************/
   /* Rule Purpose: Display initialization times of active CICS */
   /*                regions when requested. Obtain this info    */
   /*                via any GLVTEMP1.UPTIME global variable.    */
   /* Invoked when a TSO users issues OPSREQ CICINIT            */
   /*************************************************************/
   ACTREGIONS = OPSVALUE('GLVTEMP1.UPTIME','L')
   IF ACTREGIONS = '0' THEN
       SAY 'NO INITIALIZED REGIONS'
     ELSE
       DO ACTREGIONS
         PULL REGION
         UPTIME = OPSVALUE('GLVTEMP1.UPTIME.'REGION,'O')
         SAY 'CICSINIT – 'REGION' INIT TIME = 'UPTIME
       END
   RETURN
```



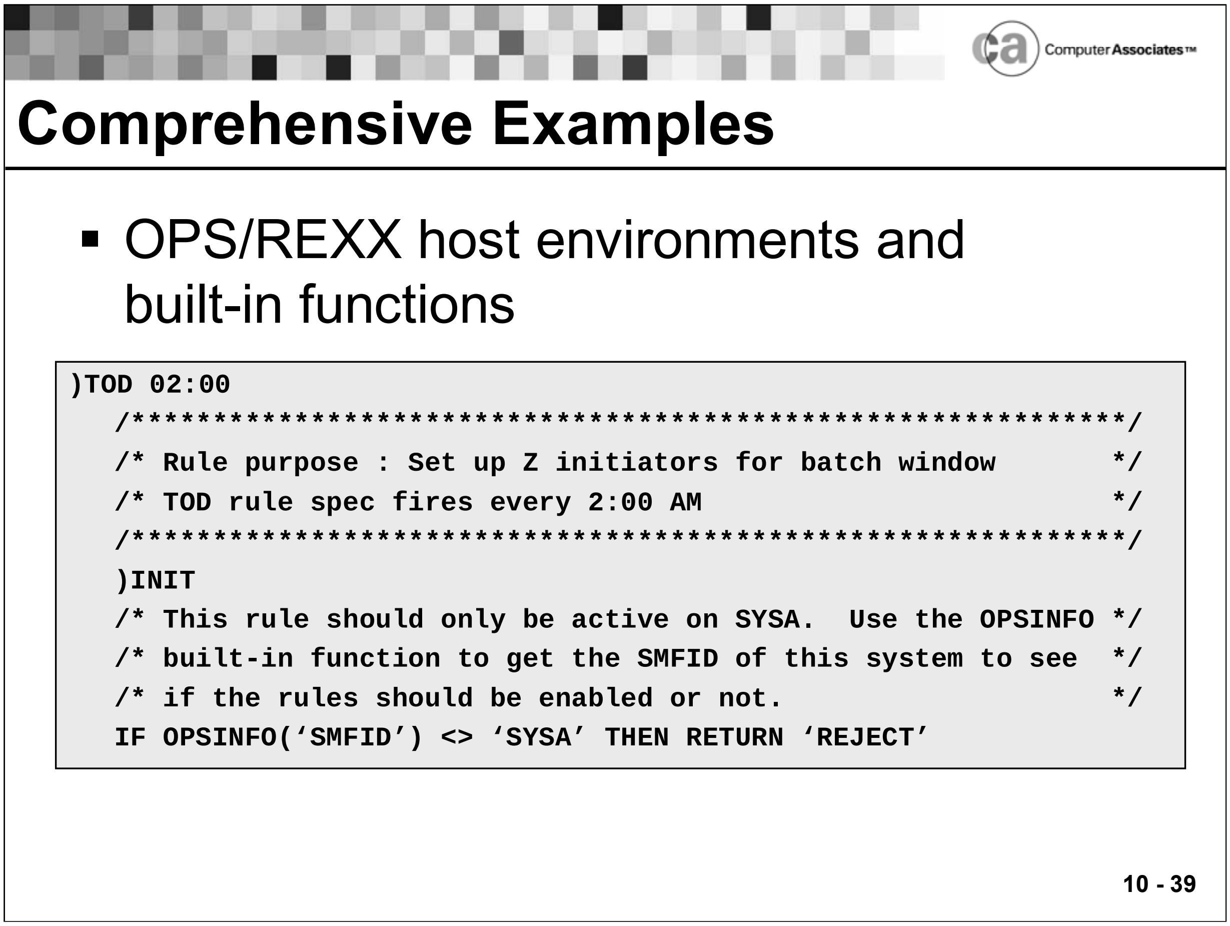
OPSVALUE Examples (continued)

- Example 3

```
/* Delete all subnodes */
 val = OPSVALUE('GLOBALA.TEST.','D')
```

- Example 4

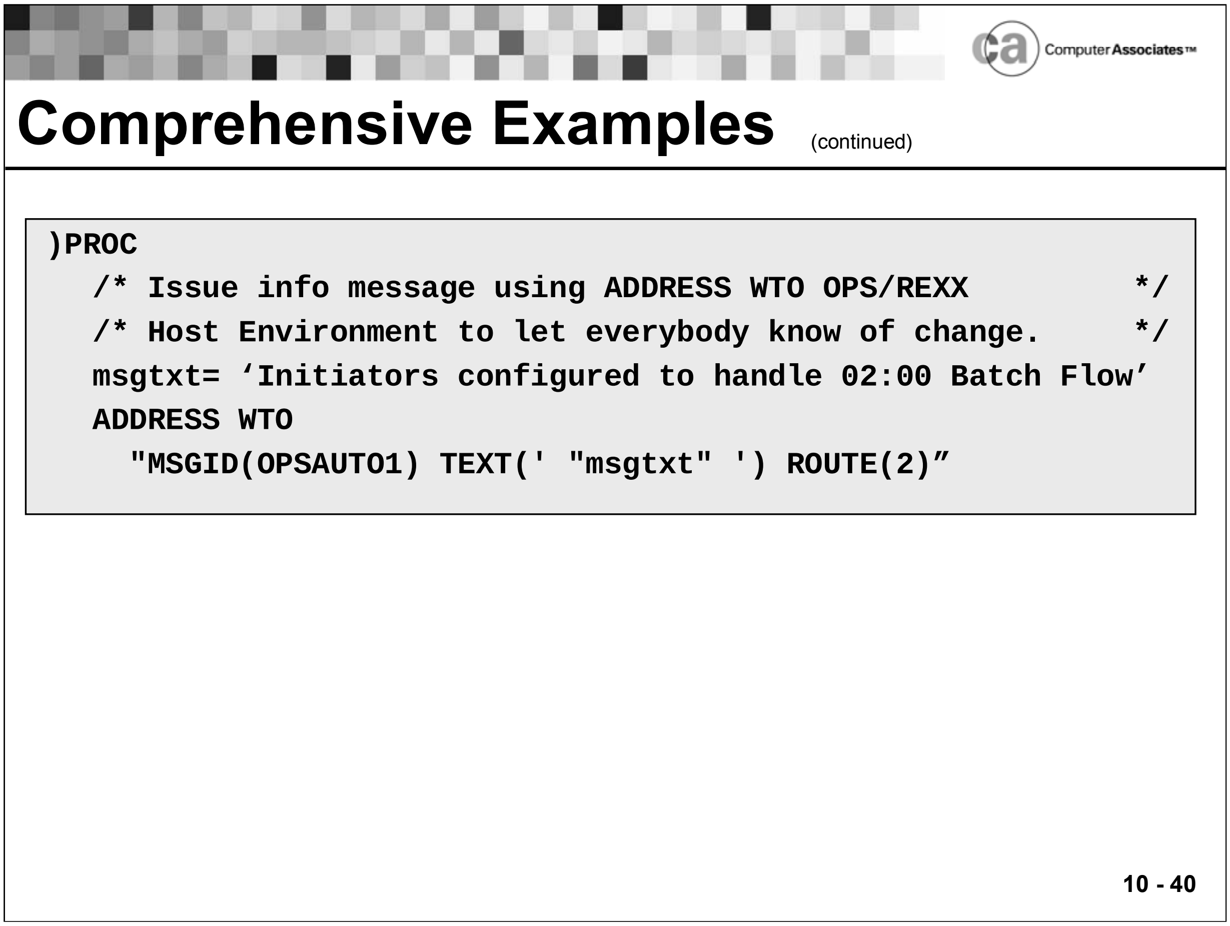
```
IF OPSVALUE('GLVJOBID.WTOCNTR','E') =
  'N' THEN GLVJOBID.WTOCNTR=0
```



Comprehensive Examples

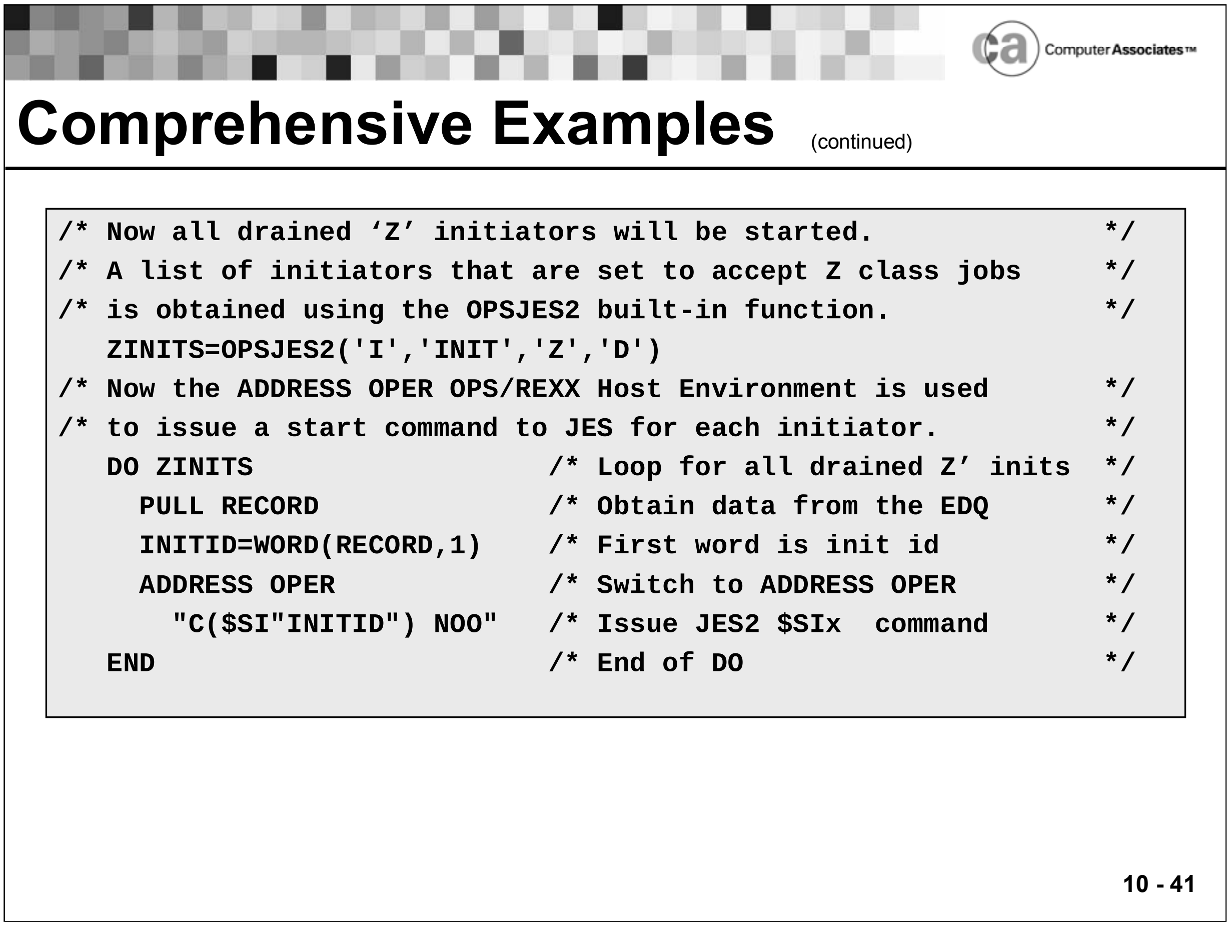
- OPS/REXX host environments and built-in functions

```
)TOD 02:00
   /*************************************************************/
   /* Rule purpose : Set up Z initiators for batch window       */
   /* TOD rule spec fires every 2:00 AM                         */
   /*************************************************************/
   )INIT
   /* This rule should only be active on SYSA.  Use the OPSINFO */
   /* built-in function to get the SMFID of this system to see  */
   /* if the rules should be enabled or not.                    */
   IF OPSINFO(‘SMFID’) <> ‘SYSA’ THEN RETURN ‘REJECT’
```



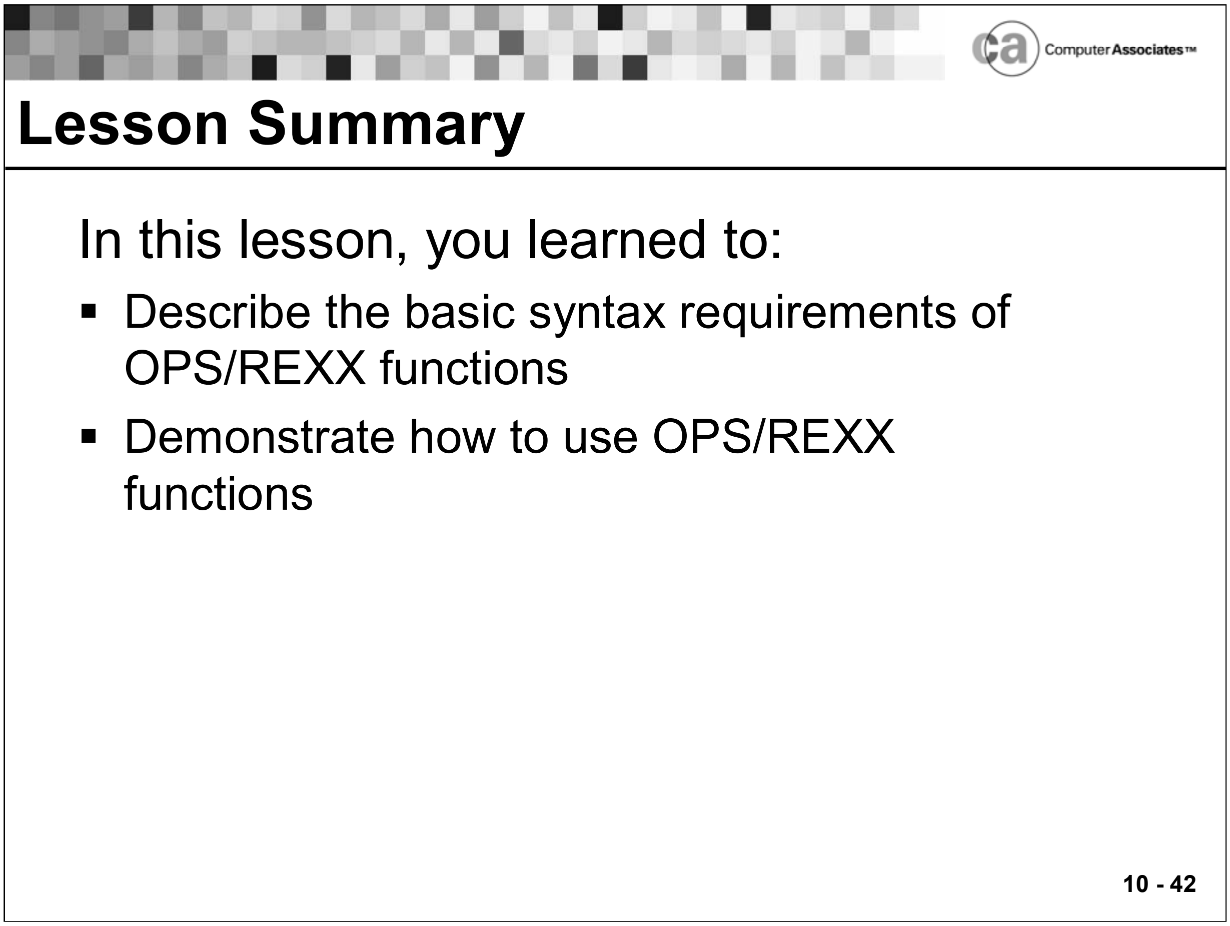
Comprehensive Examples (continued)

```
)PROC
  /* Issue info message using ADDRESS WTO OPS/REXX         */
  /* Host Environment to let everybody know of change.     */
  msgtxt= ‘Initiators configured to handle 02:00 Batch Flow’
  ADDRESS WTO
    "MSGID(OPSAUTO1) TEXT(' "msgtxt" ') ROUTE(2)”
```



Comprehensive Examples (continued)

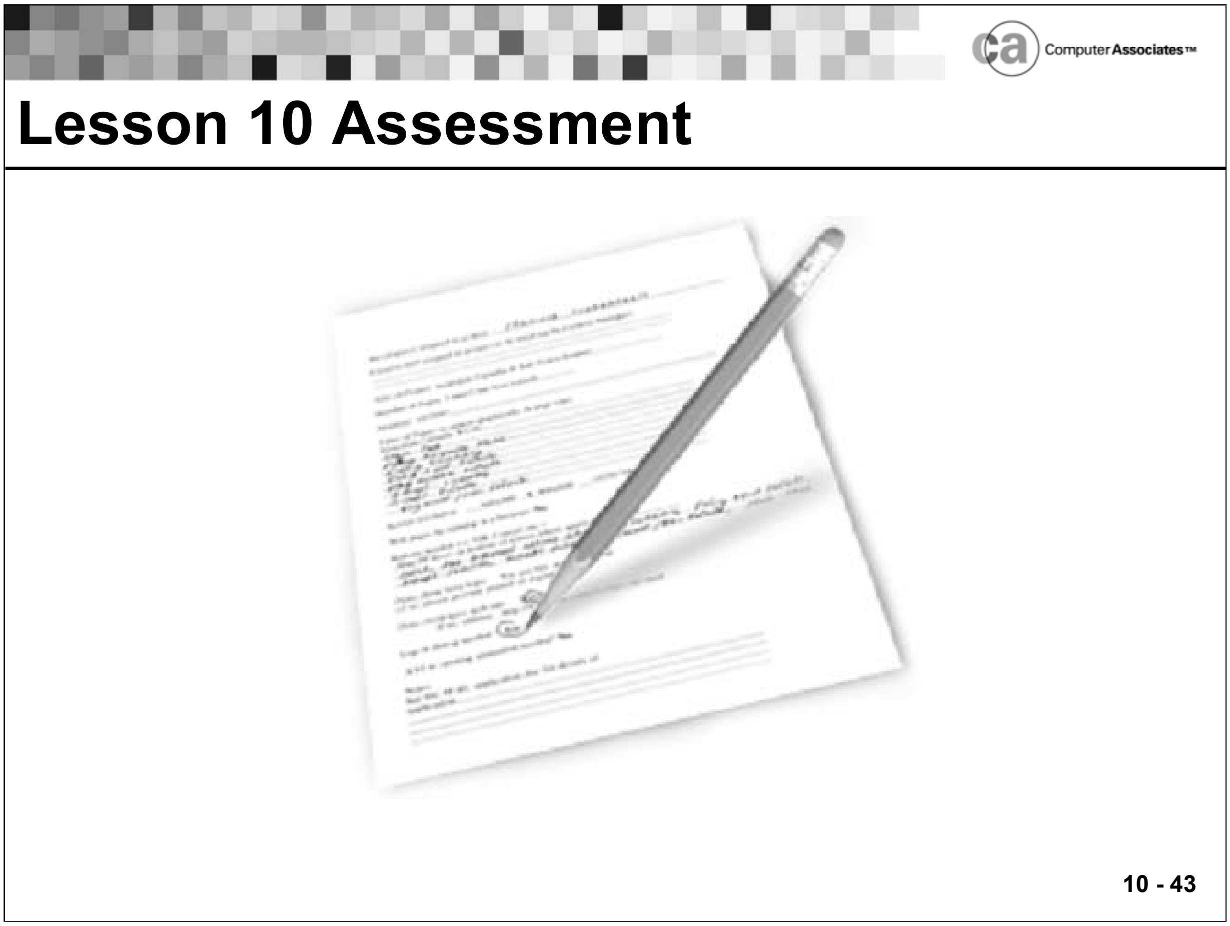
```
/* Now all drained ‘Z’ initiators will be started.              */
/* A list of initiators that are set to accept Z class jobs     */
/* is obtained using the OPSJES2 built-in function.             */
   ZINITS=OPSJES2('I','INIT','Z','D')
/* Now the ADDRESS OPER OPS/REXX Host Environment is used       */
/* to issue a start command to JES for each initiator.          */
   DO ZINITS                  /* Loop for all drained Z’ inits  */
     PULL RECORD              /* Obtain data from the EDQ       */
     INITID=WORD(RECORD,1)    /* First word is init id          */
     ADDRESS OPER             /* Switch to ADDRESS OPER         */
       "C($SI"INITID") NOO"   /* Issue JES2 $SIx  command       */
   END                        /* End of DO                      */
```



Lesson Summary

In this lesson, you learned to:

- Describe the basic syntax requirements of OPS/REXX functions
- Demonstrate how to use OPS/REXX functions



Lesson 10 Assessment

10 - 43

Lesson 10 Assessment

Write down your explanations of the following rule snippets in the space provided.

1. VOLUMES=OPSDEV('V','SYS*')

2. ISITUP=OPSTATUS('A','A','VTAMA')

3. RTVL=OPSVALUE('GLOBAL.VTAM.STAT','E')

4. DETAILS=OPSTATUS('A','I','VTAMA')

Lesson 10 Assessment (continued)

5. DEVICES=OPSDEV('U','157')

6. SAY 'ASID='OPSINFO('ASID')

7. GETWTORS=OPSTATUS('R','I','*')

8. RTVL=OPSVALUE('GLOBAL.JOB.CNT','O')

9. DEVICES=OPSDEV('D','*')

10. SAY 'SUBSYS='OPSINFO('SUBSYS')

11. RTVL=OPSVALUE('GLOBAL.ACTIVE.CICS','L')

12. SAY 'EVENTTYPE='OPSINFO('EVENTTYPE')

13. RTVL=OPSVALUE('GLOBAL.FLAG.IMSABEND','U','S0C4')

14. DEVICES=OPSDEV('D','DASD')

Lesson 10 Activity

10 - 45

Lesson 10 Activity – Using OPS/REXX Functions

In this activity, you will invoke OPS/REXX functions and see the data that they return.

Note: This activity only demonstrates the data that is returned by functions that you can use in your AOF rules.

Perform the following steps:

1. Access OPSVIEW option 2.4.
2. Specify a data set and member. (This places you in ISPF EDIT mode.)

Lesson 10 Activity (continued)

3. Type in the following code:

```
INVOKE_FUNCTION = function('x','x','x')
SAY '***VALUE OF INVOKE_FUNCTION = 'INVOKE_FUNCTION
IF QUEUED() = 0 THEN
    SAY '***FUNCTION RETURNS NO EDQ DATA'
ELSE
    SAY 'Start of captured data in the EDQ:'
    DO I = 1 TO QUEUED()
    PULL EDQ
    SAY EDQ
END
SAY '****END OF CAPTURED DATA*****'
```

4. Type OPSTATUS('A', 'I', 'JES2') in place of function('x','x', 'x') in the above code. Tab to the command line and type !OI and then press Enter. (This executes your REXX EXEC and shows you the output.) Press Enter until you are returned to ISPF EDIT mode. This function returns information about the JES2 address space to the external data queue.

5. Type OPSDEV('V', 'SYS*') in place of function('x','x', 'x') in the above code. Tab to the command line and type !OI and then press Enter. Press Enter until you are returned to ISPF EDIT mode. This function returns information about all SYS* volumes to the external data queue.

6. Type OPSJES2('I', 'INIT') in place of function('x','x', 'x') in the above code. Tab to the command line and type !OI and then press Enter. Press Enter until you are returned to ISPF EDIT mode. This function returns information about all initiators to the external data queue.

7. Type OPSJES2('I', 'INIT', 'A') in place of function('x','x', 'x') in the above code. Tab to the command line and type !OI and then press Enter. Press Enter until you are returned to ISPF EDIT mode. This function returns information about only the active initiators to the external data queue.

Lesson 10 Activity (continued)

8. Type OPSTATUS('R', 'I', '*') in place of function('x','x', 'x') in the above code. Tab to the command line and type !OI and then press Enter. Press Enter until you are returned to ISPF EDIT mode. This function returns all outstanding WTORs to the external data queue.

9. Type OPSYSSYM('I') in place of function('x','x', 'x') in the above code. Tab to the command line and type !OI and then press Enter. Press Enter until you are returned to ISPF EDIT mode. This function returns all OS/390 system symbols to the external data queue.

Final Activity

1. Access OPSVIEW option 4.5.1. Using your TSO user ID as the rule name, create a rule from scratch using the rule set given to you by your instructor. This rule should do the following:
   - Executes on the occurrence of $ HASP373 message events.
   - Be active on all systems except system XABC.
   - If the message originated from your assigned TSO user ID, suppress the message.
   - Be sure to include comments that document the date of creation, author, purpose of the rule, logic for the rule, and any user-defined variables. Test your rule using the AOF test facility.
   - Tips:

     OPSINFO('SMFID')

     MSG.JOBNAME

2. Create a second rule from scratch that does the following:
   - Executes on the occurrence of $HASP100 message events.
   - Is active on all systems except system XABC.
   - If the message originated from your assigned TSO user ID, suppress the message and remove it from the SYSLOG.
   - Be sure to include comments that document the date of creation, author, purpose of the rule, logic for the rule, and any user-defined variables. Test your rule using the AOF test facility.
   - Tips:

     OPSINFO('SMFID')

     MSG.JOBNAME

3. Create a third rule from scratch that incorporates the logic that was implemented when you created the rules in steps 1 and 2.
   - Include comments that document the date of creation, author, purpose of the rule, logic for the rule, and any user-defined variables. Test your rule using the AOF test facility.
   - Tips:

     $HASP*

     MSG.ID

Final Activity (continued)

4. Modify the rule you created in step 3 to do the following:
   - If the message is $HASP373, issue a WTO to the local master console with the message text of your choice.
   - Issue a D T z/OS command, regardless of the message event that triggers the rule.
   - Include comments that document the new logic.
   - Tips:

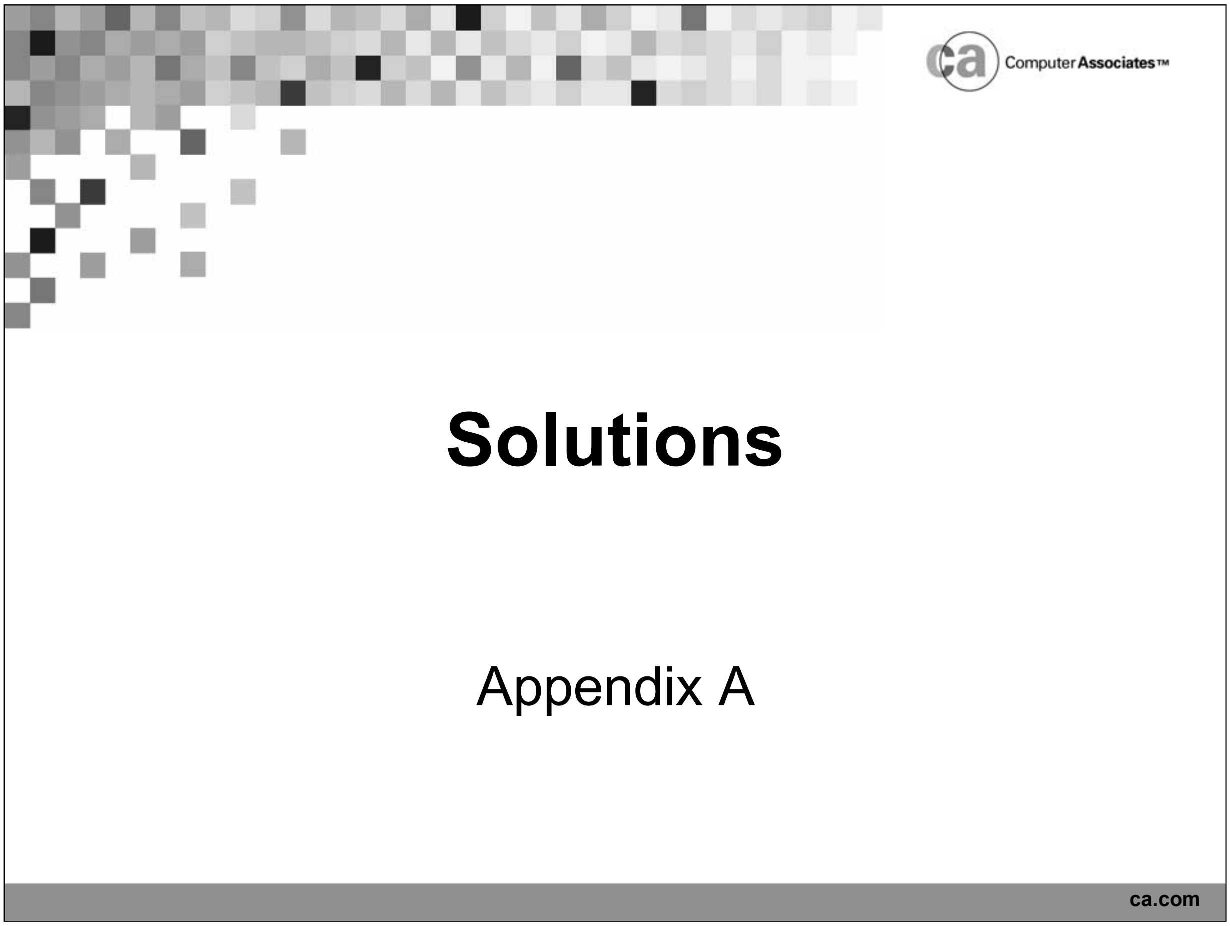
     DO…END

     OPSINFO('LOCMSTCONSNM')

     ADDRESS WTO

     ADDRESS OPER

Notes:



Solutions

Appendix A

ca.com

Lesson 1 Assessment

Match the descriptions below with one of the following terms:

OPSVIEW
SYSLOG
COF
POI
RDF
CICS
OPSLOG
Automation Point
AOF
EasyRule
OPS/REXX
REXX
Unicenter CA-OPS/MVS
Unicenter
CA-SYSVIEW/E
EPI

OPSLOG	Unicenter CA-OPS/MVS repository of system events.
EasyRule	A facility for creating rules.
Automation Point	CA's distributed automation offering.
OPS/REXX	SAA-compliant programming language used in Unicenter CA-OPS/MVS.
OPSVIEW	Operations interface for Unicenter CA-OPS/MVS.
UNICENTER CA-OPS/MVS	Automated systems operations product for z/OS environments.
EPI	Enables communications with VTAM applications.
COF	Unicenter CA-OPS/MVS interface to CICS.
RDF	A facility that lets you use SQL statements to manage data.
AOF	The heart of Unicenter CA-OPS/MVS.

Lesson 2 Assessment

1. If you wanted to view all of the events that occurred in your z/OS system, which OPSVIEW option would you select?

   Option 1 – OPSLOG.

2. What would you type to “jump” to the Automation Analyzer Specification panel?

   =7.2

3. Is there another way that you can access the Automation Analyzer Specification panel? If so, explain how.

   Yes. Type 2 in the Option field on the OPSVIEW Utilities menu.

4. Which command would you use to view a list of all possible data columns in the OPSLOG?

   DISPLAY.

5. Which facility would you use if you wanted to generate rules but you had little or no REXX experience?

   EasyRule (OPSVIEW Option 2.3).

6. Where does the Automation Analyzer obtain the data that it uses in its analysis of the automation practices at your site?

   OPSLOG

7. Can you enter z/OS commands via OPSVIEW?

   Yes.

8. What would you type to “jump” to the EasyRule facility?

   =2.3

9. Can you invoke EasyRule from the Automation Analyzer?

   Yes.

10. Describe the difference between OPSVIEW and Unicenter CA-SYSVIEW.

    OPSVIEW is the operations interface for Unicenter CA-OPS/MVS. Unicenter CA-SYSVIEW is a performance monitor.

Lesson 3 Assessment

Note: A week runs from Sunday to Saturday.

1. True/False: AOF rules contain OPS/REXX programs.

   True.

2. When would this rule execute?)TOD 06:00,,,6

   Every day at 6:00 for the next 6 days.

3. What is the major function of the AOF?

   To monitor system events and respond to them.

4. Which rule section specifies the actions the rule takes when it becomes enabled?

   Initialization.

5. True/False: Each rule set is a separate data set.

   True.

6. When would this rule execute?)TOD 07:30,,,,,CATCHUPYES

   Every day at 7:30. It will catch up if Unicenter CA-OPS/MVS is down.

7. Name three methods for writing rules.

   TSO EDIT, ISPF/PDF EDIT, OPSVIEW's Edit option, EasyRule.

8. Which rule type can be used as an effective tool for operators to perform the necessary command sequences for shift changes?

   Pseudo command (CMD).

9. Which rule section specifies the system event that causes the rule to execute?

   Event definition.

10. True/False: A rule must contain an)END statement.

    False.

Lesson 3 Assessment (continued)

11. How does the AOF recognize and respond to events?

    Rules.

12. Which rule type enables you to create a new command?

    Command (CMD)

13. When would this rule execute?)TOD 02:00

    Every day at 2:00.

14. Which rule section specifies the action the rule takes when it is disabled?

    Termination.

15. True/False: Rules always execute in a predictable order.

    True.

16. Which part of a rule marks the end of the rule?

    END statement.

17. Which rule section specifies the action to take in response to the system event that triggered the rule?

    Processing.

18. True/False: Rules are stored within rule sets.

    True.

19. Which rule type enables you to respond to a WTOR message?

    Message (MSG).

20. When would this rule execute?)TOD *+2 MINUTES, 15 SECONDS,,5

    Two minutes after it is enabled. Then every 15 seconds until it executes a total of 5 times.

Lesson 5 Activity – Solving a Problem With EasyRule

Scenario

- You need to be aware of NOT CATALOGED 2 conditions because they are indicative of production problems. These conditions are identified in IEF287I messages.
- IEF285I messages are related normal messages that do not require action.
- You want to increase the visibility of the NOT CATALOGED 2 conditions. The presence of IEF285I messages creates visual noise, making it difficult for you to see and respond to the important IEF287I messages.

Task

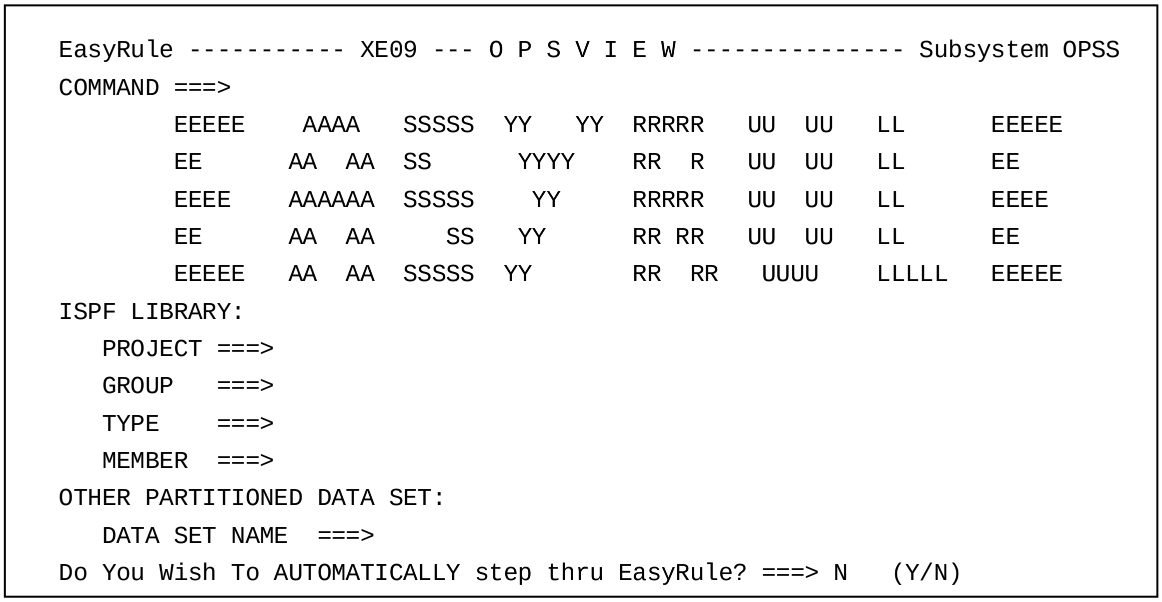
- Write two rules.

  The first rule, called MNSTATUS, should not only suppress IEF285I messages from displaying on the console, but also keep them from appearing in the SYSLOG.

  The second rule, called NOTCTLG, should highlight IEF287I messages and post messages as described to the job log.

Steps to take:

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:



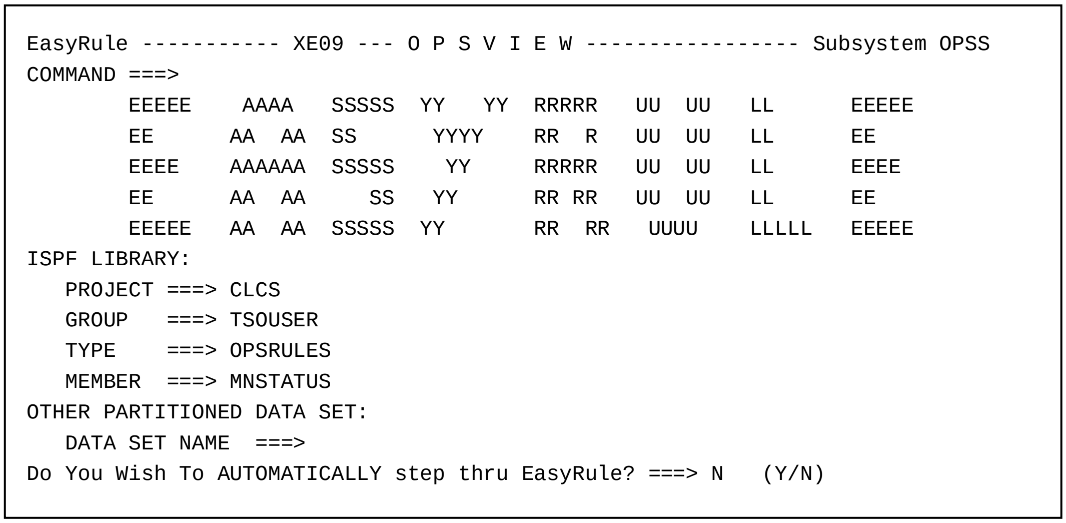
```
EasyRule ----------- XE09 --- O P S V I E W --------------- Subsystem OPSS
COMMAND ===>
        EEEEE    AAAA   SSSSS  YY   YY  RRRRR   UU  UU   LL      EEEEE
        EE      AA  AA  SS      YYYY    RR  R   UU  UU   LL      EE
        EEEE    AAAAAA  SSSSS    YY     RRRRR   UU  UU   LL      EEEE
        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)
```

Lesson 5 Activity (continued)

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 EasyRule Primary panel's Project, Group, and Type fields 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:

   - Specify the data set that is to contain the new rule. For example, this solution uses the CLCS.TSOUSER.OPSRULES data set. Type *CLCS* in the Project field, *TSOUSER* in the Group field, and *OPSRULES* in the Type field. (In place of *TSOUSER*, type the user ID given to you by your instructor.)
   - Name the rule MNSTATUS. Type MNSTATUS in the Member field.
   - *Do not* select automatic step-through for panel navigation. Instead, you will use EasyRule's 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:

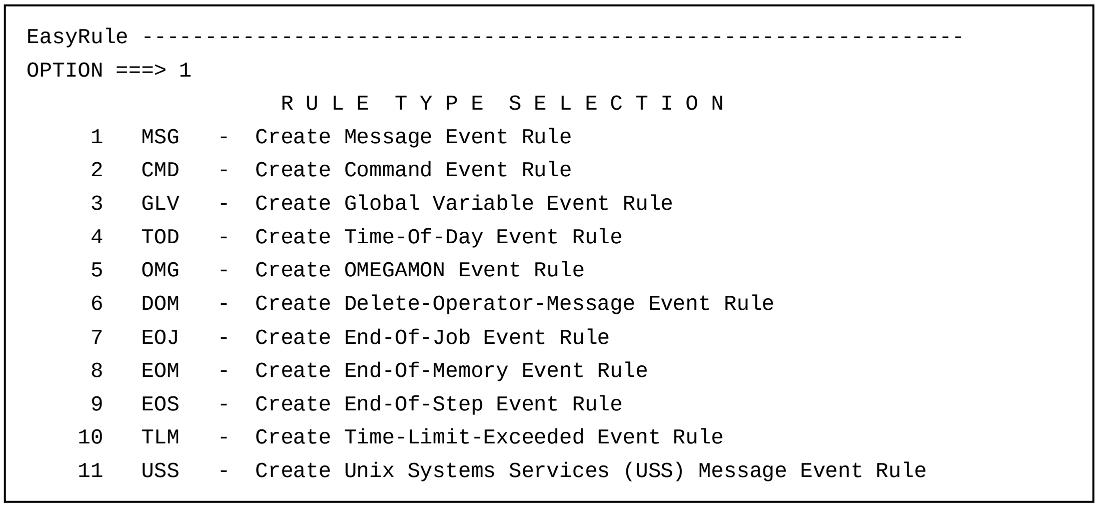


```
EasyRule ----------- XE09 --- O P S V I E W ----------------- Subsystem OPSS
COMMAND ===>
        EEEEE    AAAA   SSSSS  YY   YY  RRRRR   UU  UU   LL      EEEEE
        EE      AA  AA  SS      YYYY    RR  R   UU  UU   LL      EE
        EEEE    AAAAAA  SSSSS    YY     RRRRR   UU  UU   LL      EEEE
        EE      AA  AA     SS   YY      RR RR   UU  UU   LL      EE
        EEEEE   AA  AA  SSSSS  YY       RR  RR   UUUU    LLLLL   EEEEE
ISPF LIBRARY:
   PROJECT ===> CLCS
   GROUP   ===> TSOUSER
   TYPE    ===> OPSRULES
   MEMBER  ===> MNSTATUS
OTHER PARTITIONED DATA SET:
   DATA SET NAME  ===>
Do You Wish To AUTOMATICALLY step thru EasyRule? ===> N   (Y/N)
```

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

Lesson 5 Activity (continued)

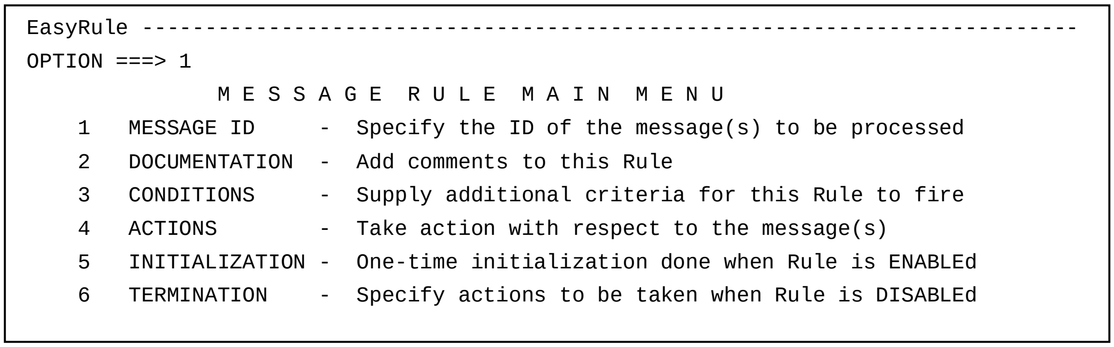
3. **Select the type of rule you want to create.** From the Rule Type Selection panel, choose option 1 to create a message rule. Notice that this has already been done in the sample panel shown below.



```
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 Systems Services (USS) Message Event Rule
```

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 sample below.

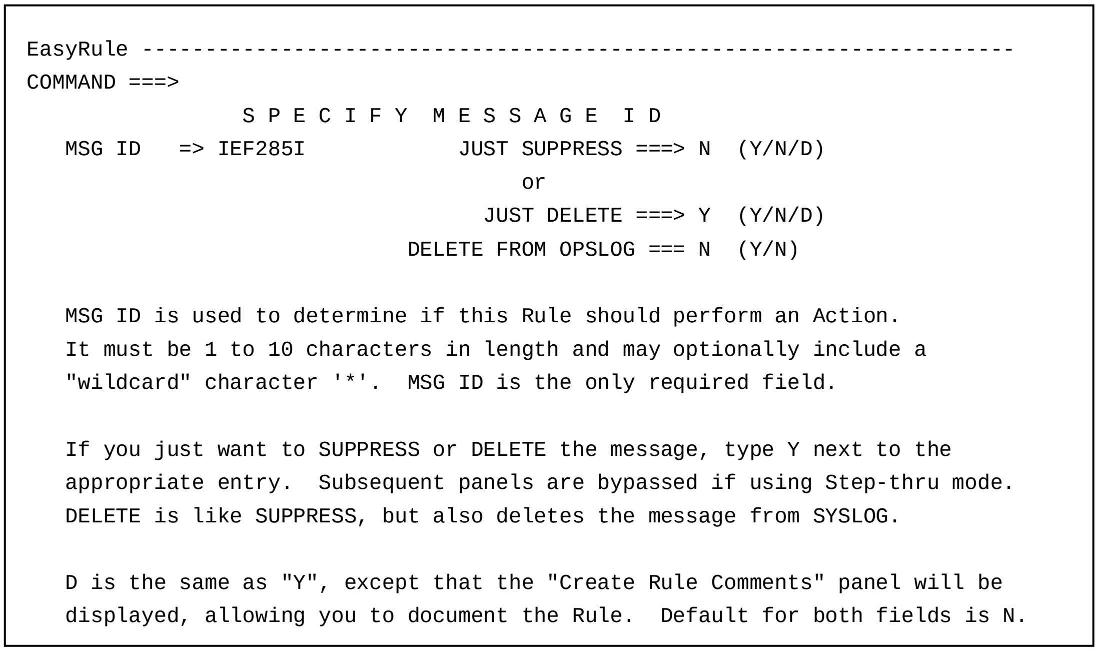


```
EasyRule --------------------------------------------------------------------------
OPTION ===> 1
               M E S S A G E   R U L E   M A I N   M E N U
    1   MESSAGE ID     -  Specify the ID of the message(s) to be processed
    2   DOCUMENTATION  -  Add comments to this Rule
    3   CONDITIONS     -  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.

Lesson 5 Activity (continued)

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 Y in the Just Delete field to indicate that you not only want to suppress the message, but also want to keep it from appearing in SYSLOG. In the sample shown below, the MSG ID and Just Delete fields have been filled in.

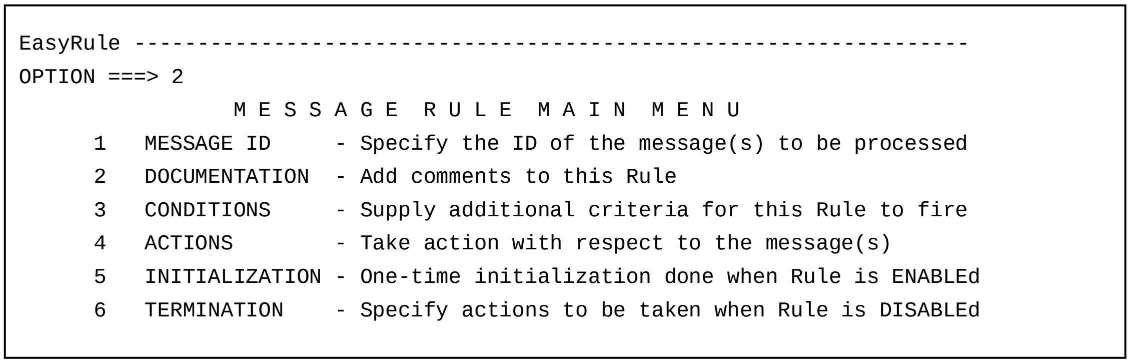


```
EasyRule ---------------------------------------------------------------------
COMMAND ===>
                 S P E C I F Y  M E S S A G E  I D
   MSG ID   => IEF285I            JUST SUPPRESS ===> N  (Y/N/D)
                                       or
                                    JUST DELETE ===> Y  (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 the "Create Rule Comments" panel will be
   displayed, allowing you to document the Rule.  Default for both fields is N.
```

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.

Lesson 5 Activity (continued)

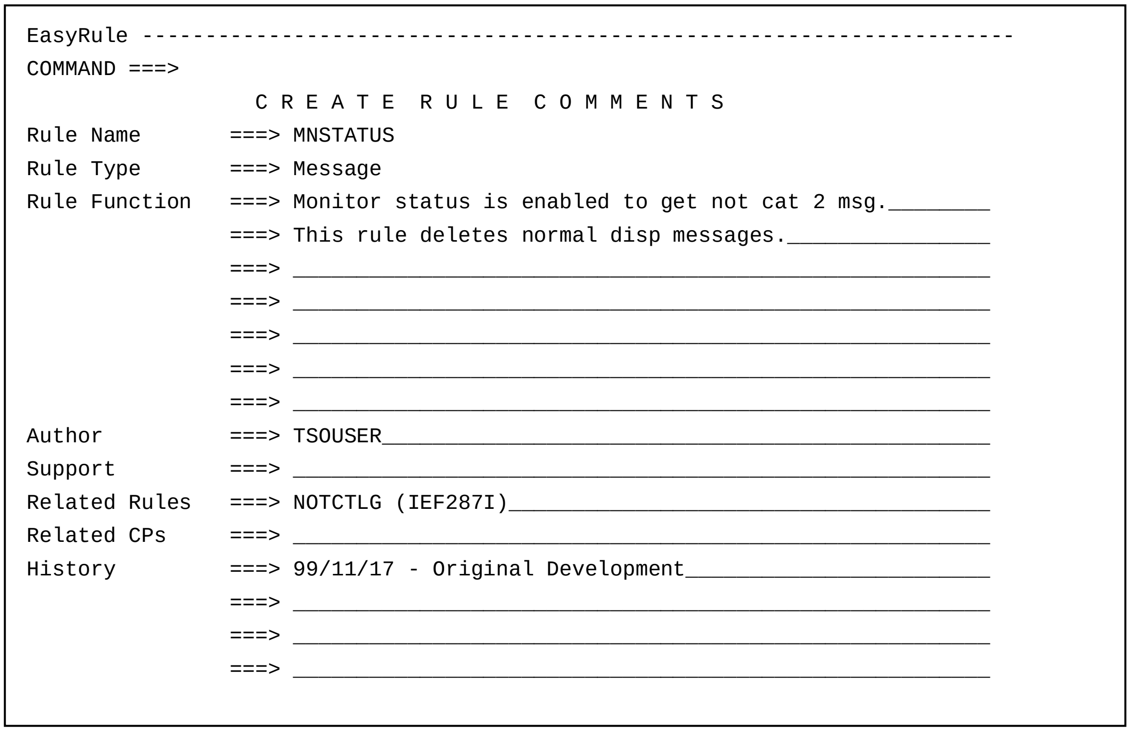
6. **Select DOCUMENTATION from the Message Rule Main Menu.** From the Message Rule Main Menu, select 2, for DOCUMENTATION. Notice that this has been done in the sample shown here:



```
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 follows.

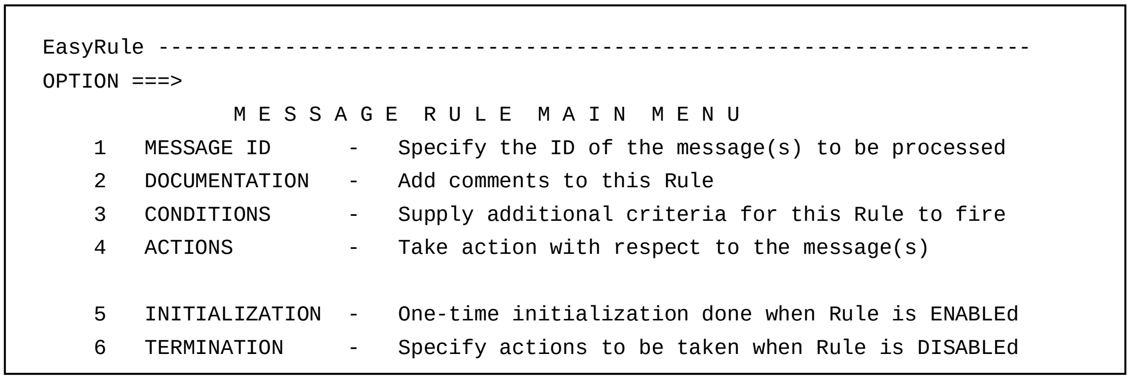
7. **Document the MNSTATUS rule.** Type comments similar to those shown below onto your own Create Rule Comments panel.



```
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          ===> TSOUSER________________________________________________
Support         ===> _______________________________________________________
Related Rules   ===> NOTCTLG (IEF287I)______________________________________
Related CPs     ===> _______________________________________________________
History         ===> 99/11/17 - Original Development________________________
                ===> _______________________________________________________
                ===> _______________________________________________________
                ===> _______________________________________________________
```

Lesson 5 Activity (continued)

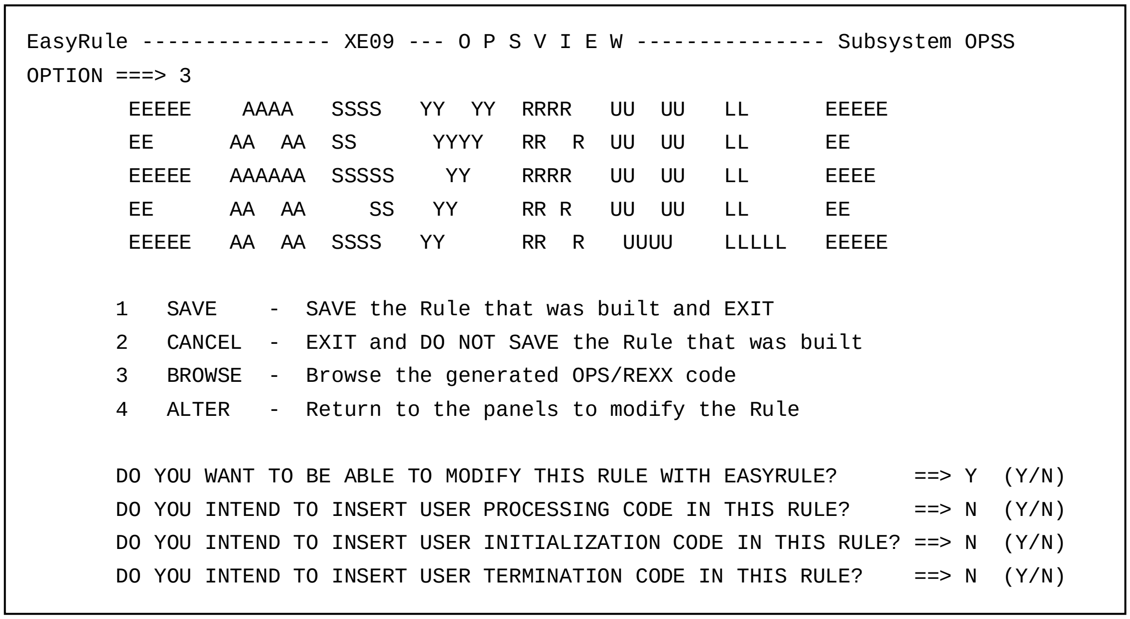
After you enter comments for the rule, EasyRule returns you to the Message Rule Main Menu (shown below).



```
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).

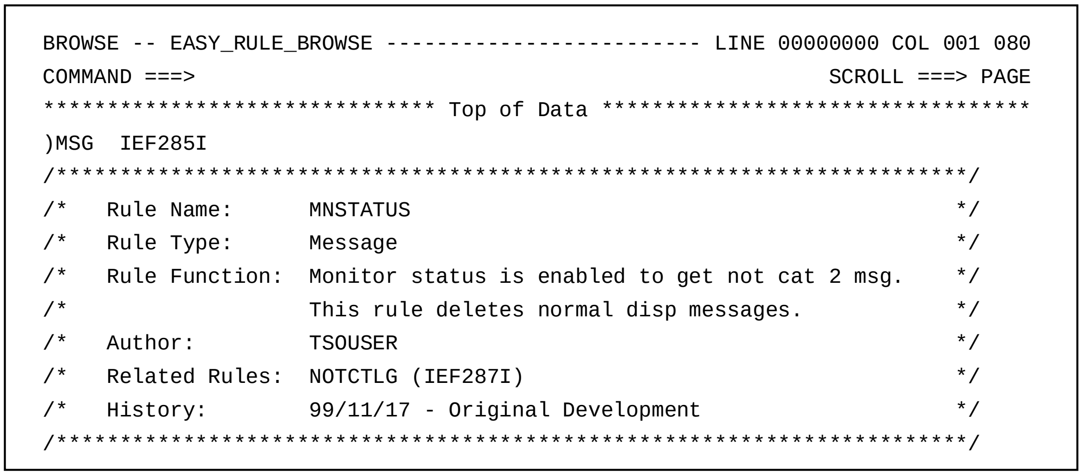
9. **Review the OPS/REXX code that EasyRule built.** On the EasyRule Final Options Menu, choose option 3, as shown here:



```
EasyRule --------------- XE09 --- O P S V I E W --------------- Subsystem OPSS
OPTION ===> 3
        EEEEE    AAAA   SSSS   YY  YY  RRRR   UU  UU   LL      EEEEE
        EE      AA  AA  SS      YYYY   RR  R  UU  UU   LL      EE
        EEEEE   AAAAAA  SSSSS    YY    RRRR   UU  UU   LL      EEEE
        EE      AA  AA     SS   YY     RR R   UU  UU   LL      EE
        EEEEE   AA  AA  SSSS   YY      RR  R   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)
```

Lesson 5 Activity (continued)

When you select option 3, a panel similar to the one shown below 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:         TSOUSER                                            */
/*   Related Rules:  NOTCTLG (IEF287I)                                  */
/*   History:        99/11/17 - Original Development                    */
/************************************************************************/
```

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.

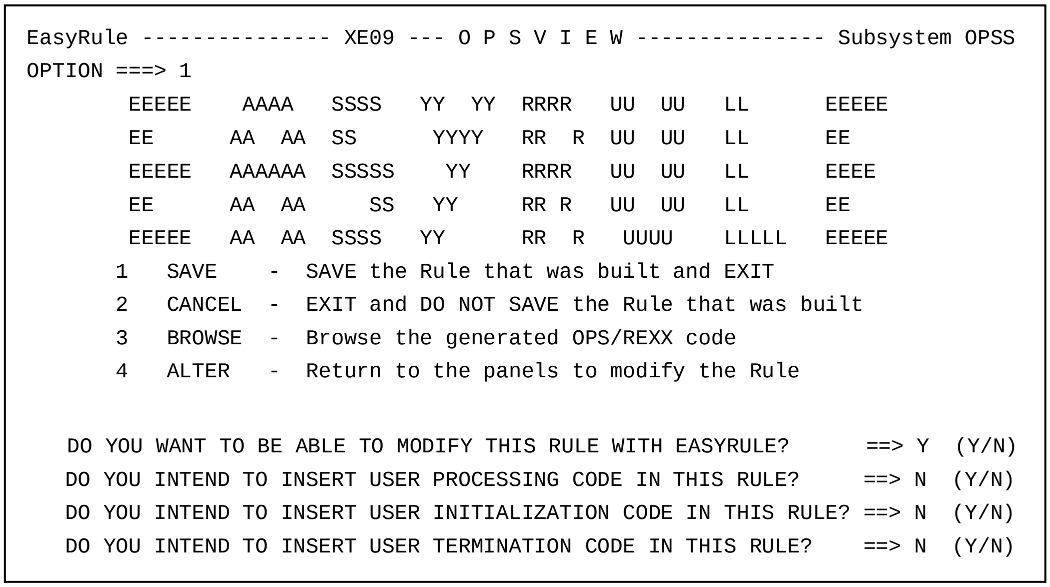
The OPS/MVS base product has the following components:

OPS/REXX Code	Panel Entries
MSG IEF285I	Rule Type Selection Panel and Specify Message ID Panel
Comments box	Create Rule Comments Panel
Return “DELETE”	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 follows.

Lesson 5 Activity (continued)

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 Option field below.



```
EasyRule --------------- XE09 --- O P S V I E W --------------- Subsystem OPSS
OPTION ===> 1
        EEEEE    AAAA   SSSS   YY  YY  RRRR   UU  UU   LL      EEEEE
        EE      AA  AA  SS      YYYY   RR  R  UU  UU   LL      EE
        EEEEE   AAAAAA  SSSSS    YY    RRRR   UU  UU   LL      EEEE
        EE      AA  AA     SS   YY     RR R   UU  UU   LL      EE
        EEEEE   AA  AA  SSSS   YY      RR  R   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)
```

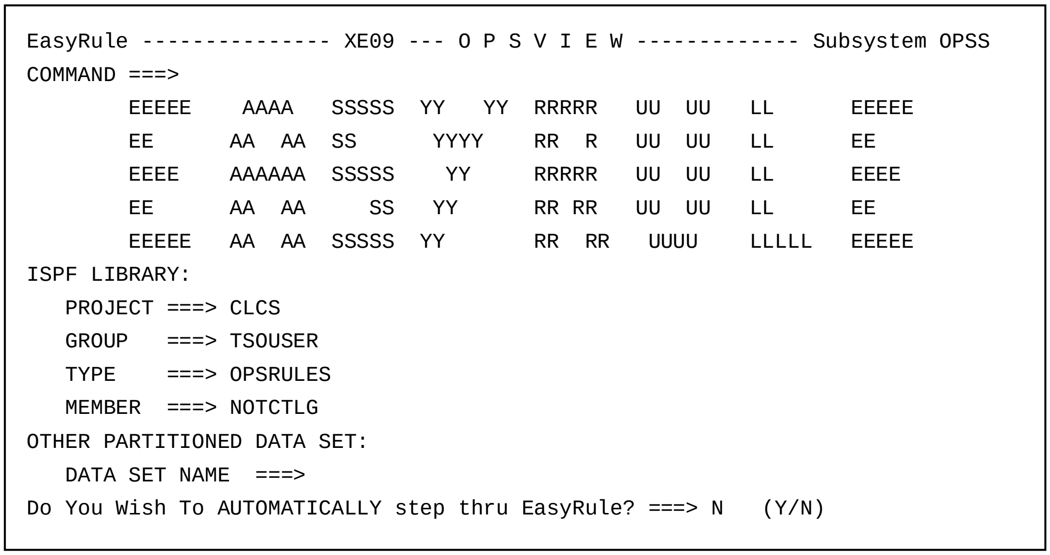
When you choose 1 from the EasyRule Final Options Menu, EasyRule saves the rule to the data set and member you specified earlier, and returns you to the EasyRule Primary panel.

11. **Specify a data set and member for the 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 CLCS.TSOUSER.OPSRULES data set to contain the MNSTATUS rule. Type *CLCS* in the Project field, *TSOUSER* in the Group field, and *OPSRULES* in the Type field. (In place of *TSOUSER*, type the user ID given to you by your instructor.)
    - Name the rule NOTCTLG. Type NOTCTLG in the Member field.
    - Once again, accept the default of N for the automatic step-through prompt.

Lesson 5 Activity (continued)

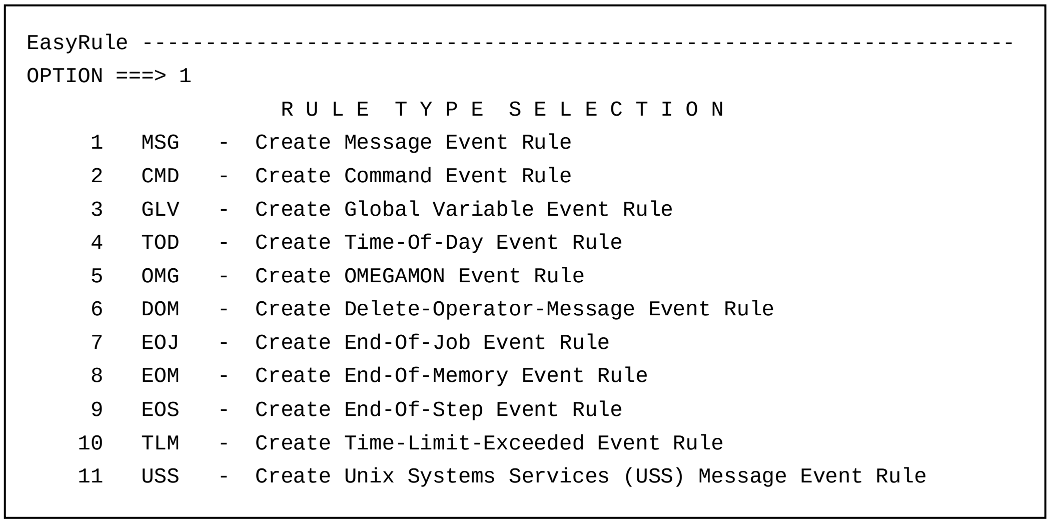
Your panel should now be similar to the one shown here:



```
EasyRule --------------- XE09 --- 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 ===> CLCS
   GROUP   ===> TSOUSER
   TYPE    ===> OPSRULES
   MEMBER  ===> NOTCTLG
OTHER PARTITIONED DATA SET:
   DATA SET NAME  ===>
Do You Wish To AUTOMATICALLY step thru EasyRule? ===> N   (Y/N)
```

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

12. **Select the type of rule you want to create.** To select a rule type, enter its code into 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:

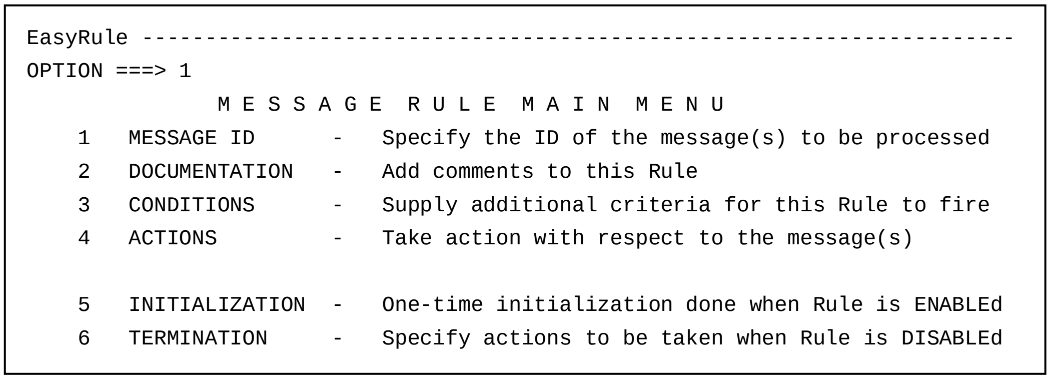


```
EasyRule ---------------------------------------------------------------------
OPTION ===> 1
                    R U L E   T Y P E   S E L E C T I O N
     1   MSG   -  Create Message Event Rule
     2   CMD   -  Create Command Event Rule
     3   GLV   -  Create Global Variable Event Rule
     4   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 Systems Services (USS) Message Event Rule
```

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.

Lesson 5 Activity (continued)

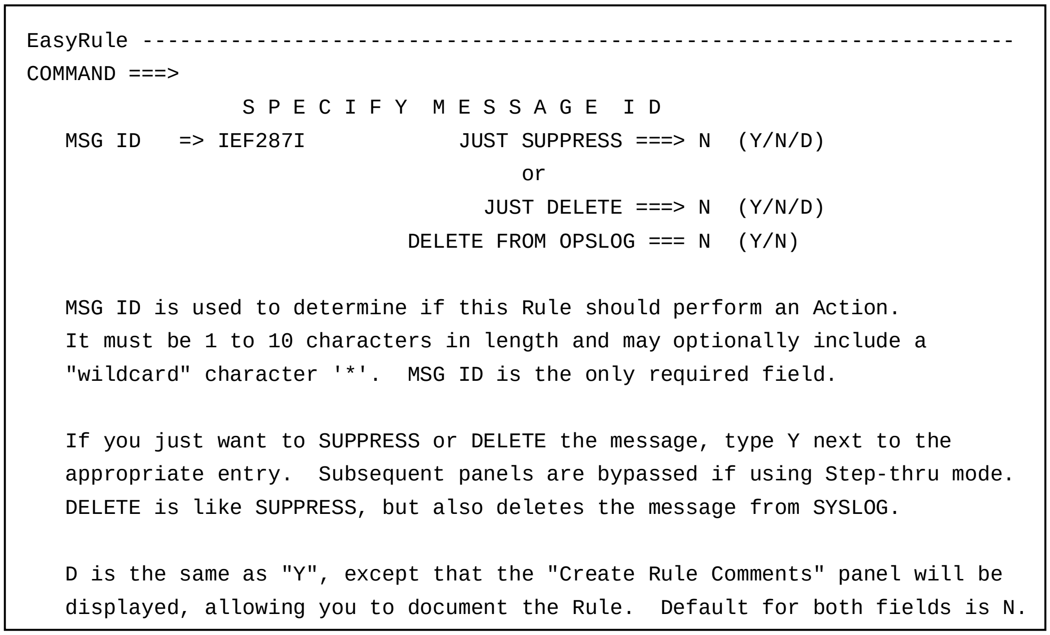
13. **Select MESSAGE ID from the Message Rule Main Menu**. Select option 1, for MESSAGE ID, from the Message Rule Main Menu. Notice that 1 has been specified in the sample panel shown here:



```
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.

14. **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:

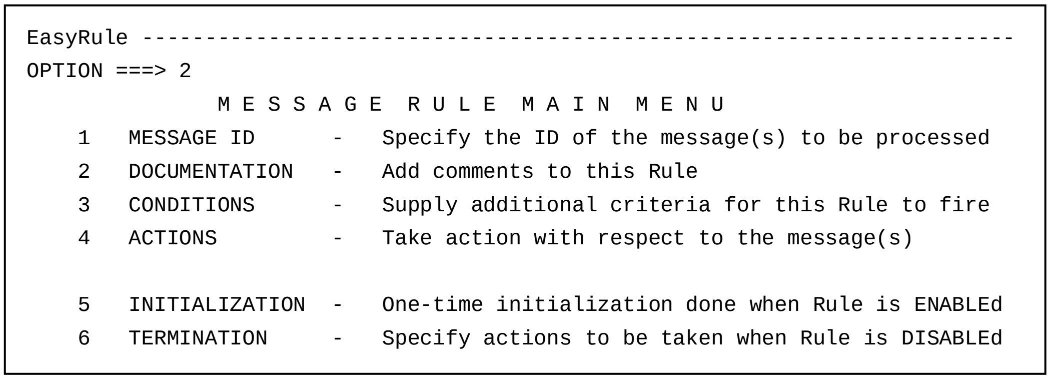


```
EasyRule ------------------------------------------------------------------
COMMAND ===>
                 S P E C I F Y  M E S S A G E  I D
   MSG ID   => IEF287I            JUST SUPPRESS ===> N  (Y/N/D)
                                       or
                                    JUST DELETE ===> 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 the "Create Rule Comments" panel will be
   displayed, allowing you to document the Rule.  Default for both fields is N.
```

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

Lesson 5 Activity (continued)

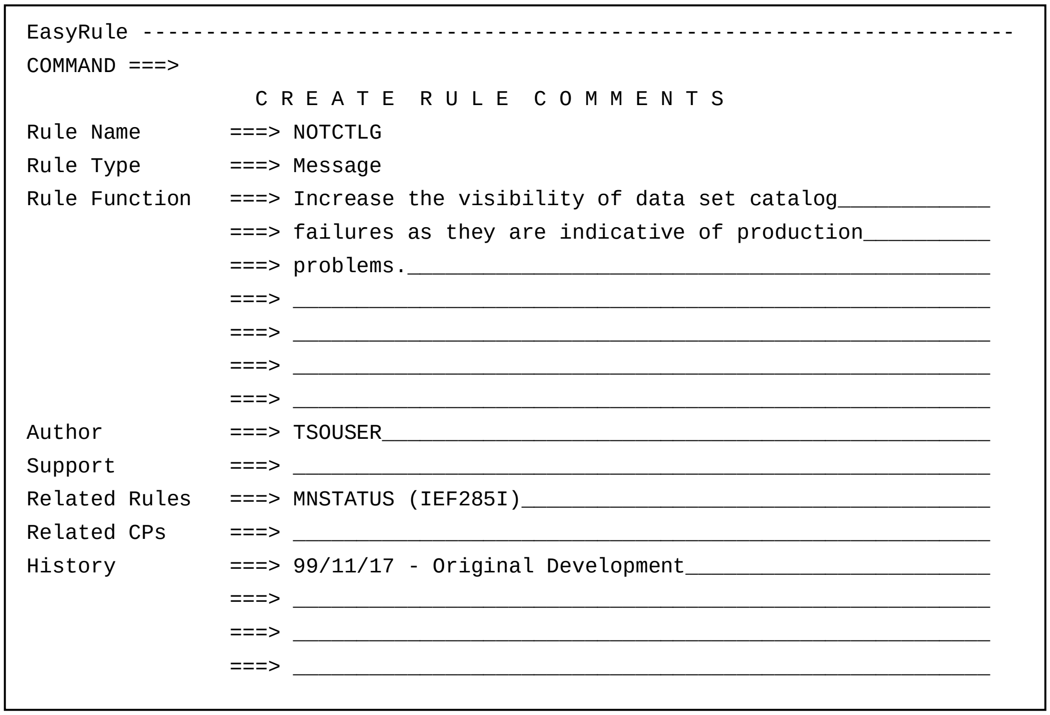
15. **Select DOCUMENTATION from the Message Rule Main Menu.** From Message Rule Main Menu, select 2, for DOCUMENTATION. Notice that 2 has been specified in the sample here:



```
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
```

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

16. **Document the rule you are creating.** It is always a good idea to provide comments for a rule. Type comments similar to those shown below onto 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 Rule Function section.

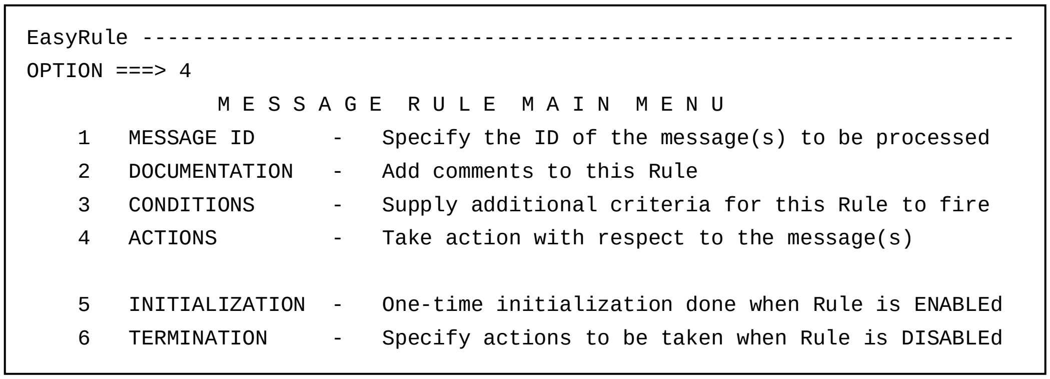


```
EasyRule --------------------------------------------------------------------
COMMAND ===>
                  C R E A T E  R U L E  C O M M E N T S
Rule Name       ===> NOTCTLG
Rule Type       ===> Message
Rule Function   ===> Increase the visibility of data set catalog____________
                ===> failures as they are indicative of production__________
                ===> problems.__________________________________________________
                ===> _______________________________________________________
                ===> _______________________________________________________
                ===> _______________________________________________________
                ===> _______________________________________________________
Author          ===> TSOUSER________________________________________________
Support         ===> _______________________________________________________
Related Rules   ===> MNSTATUS (IEF285I)_____________________________________
Related CPs     ===> _______________________________________________________
History         ===> 99/11/17 - Original Development________________________
                ===> _______________________________________________________
                ===> _______________________________________________________
                ===> _______________________________________________________
```

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

Lesson 5 Activity (continued)

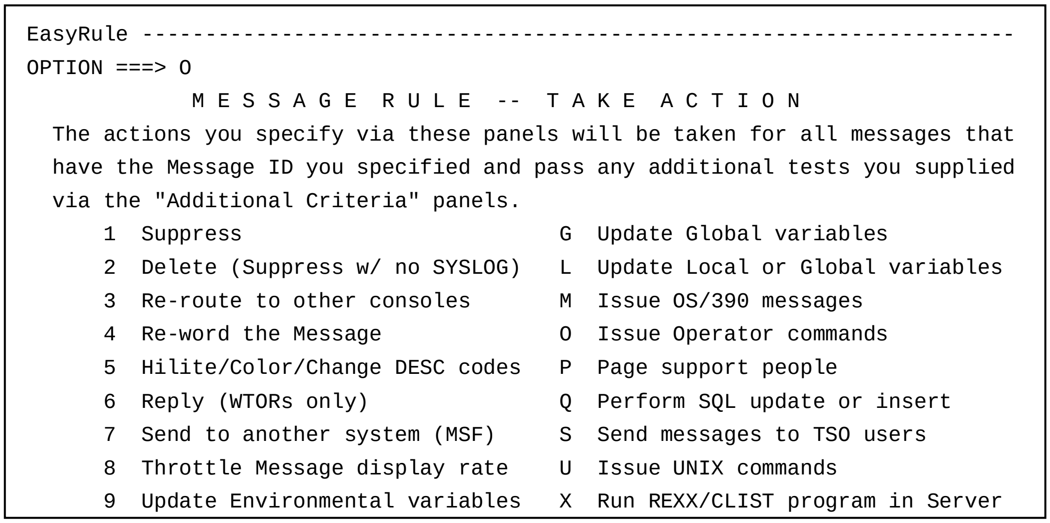
17. **Select ACTIONS from the Message Rule Main Menu.** From the Message Rule Main Menu, select 4, for ACTIONS. Notice that 4 has been specified in the sample shown here:



```
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.

18. **Select O from the Take Action menu.** For this example, suppose that you want OPS/MVS to write the IEF287I messages to your job log. To do this, you need to instruct 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 ===> O
             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  Delete (Suppress w/ no SYSLOG)   L  Update Local or Global variables
      3  Re-route to other consoles       M  Issue OS/390 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
```

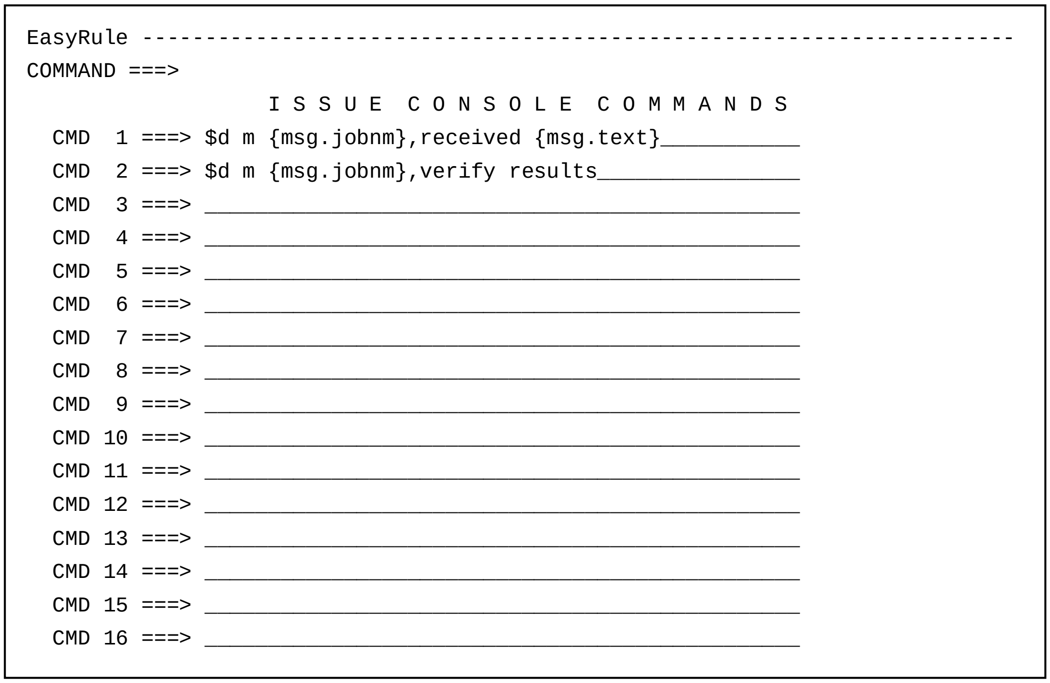
After you enter O in the Option field, the Issue Console Commands panel appears. A sample panel appears next.

Lesson 5 Activity (continued)

19. **Write messages to the job log.** You can now enter the commands that you want OPS/MVS to issue when the rule is enabled. On your Issue Console Commands panel, type the commands as shown below. These entries will cause 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.

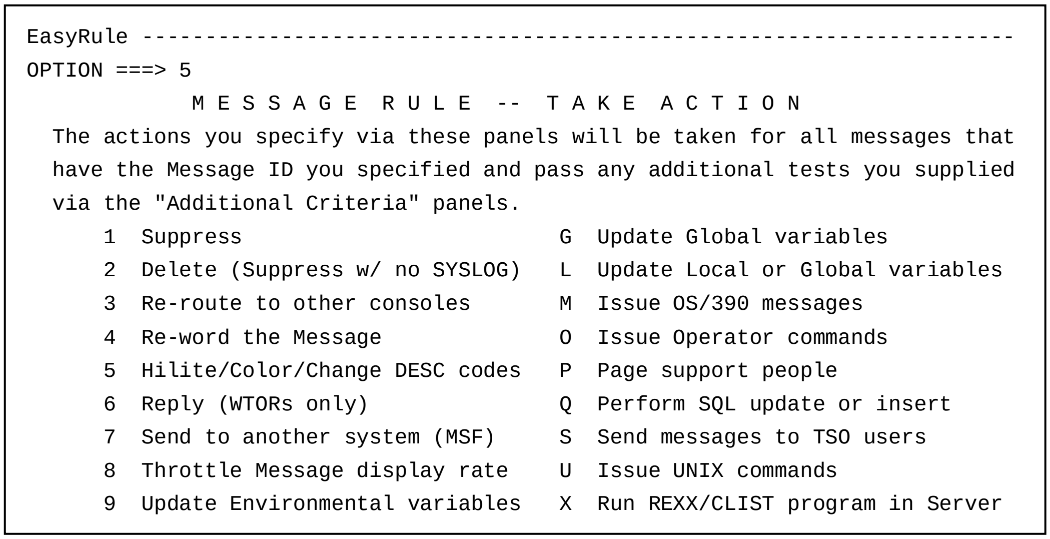


```
EasyRule -----------------------------------------------------------------------
COMMAND ===>
                   I S S U E   C O N S O L E   C O M M A N D S
  CMD  1 ===> $d m {msg.jobnm},received {msg.text}___________
  CMD  2 ===> $d m {msg.jobnm},verify results________________
  CMD  3 ===> _______________________________________________
  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 (shown next).

Lesson 5 Activity (continued)

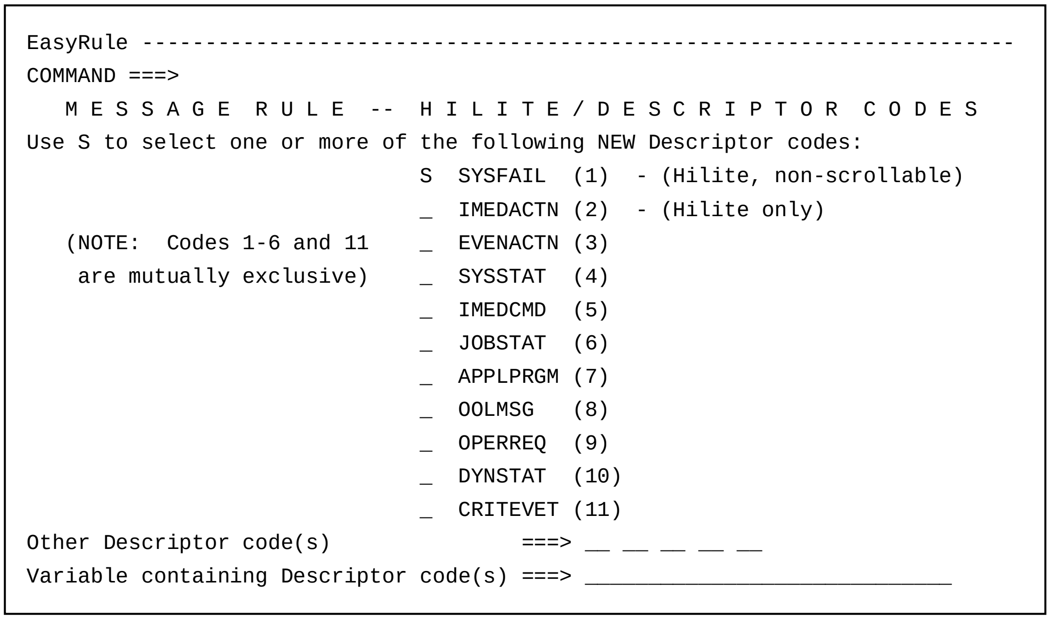
20. **Select 5 from the Take Action menu.** Select 5, for Hilite/Color/Change DESC Codes. Notice that in the sample shown below, 5 has been specified.



```
EasyRule --------------------------------------------------------------------
OPTION ===> 5
             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  Delete (Suppress w/ no SYSLOG)   L  Update Local or Global variables
      3  Re-route to other consoles       M  Issue OS/390 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
```

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

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



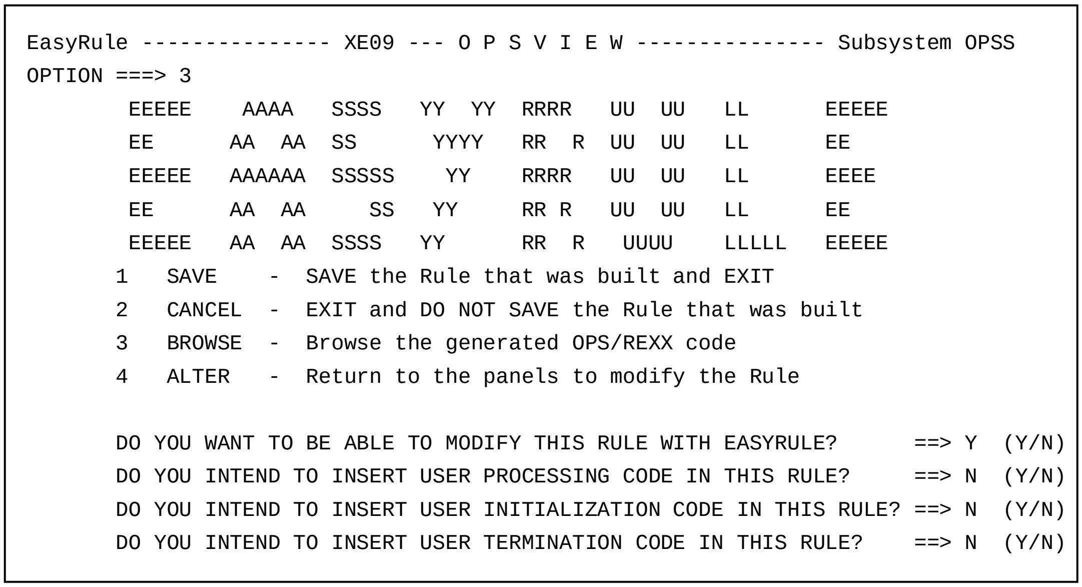
```
EasyRule --------------------------------------------------------------------
COMMAND ===>
   M E S S A G E  R U L E  --  H I L I T E / D E S C R I P T O R  C O D E S
Use S to select one or more of the following NEW Descriptor codes:
                               S  SYSFAIL  (1)  - (Hilite, non-scrollable)
                               _  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, you return to Take Action menu.

Lesson 5 Activity (continued)

22. **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.

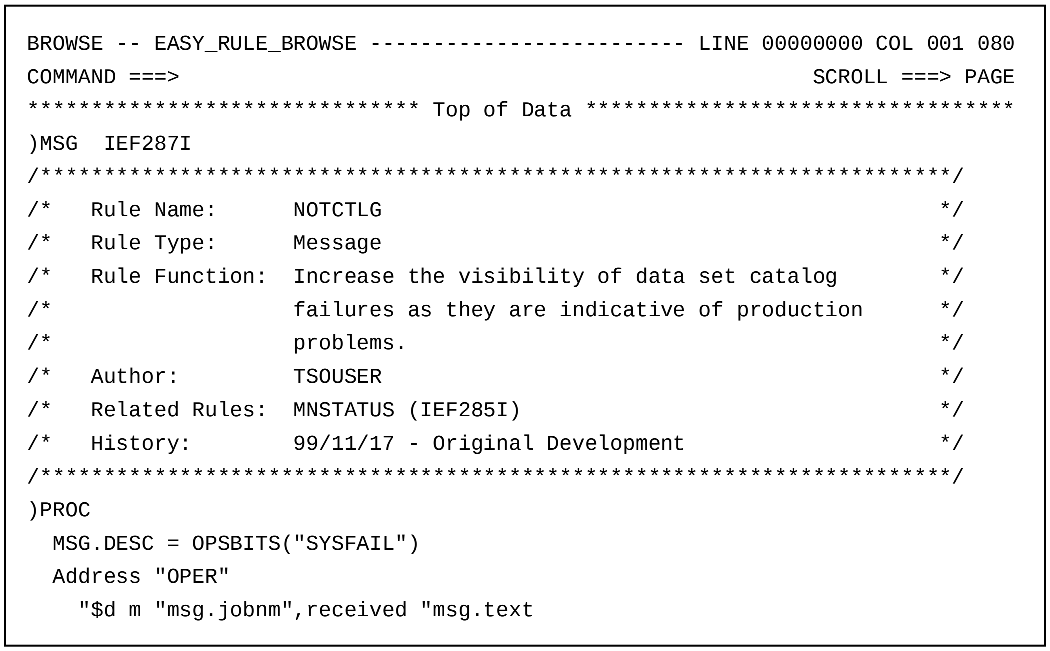
23. **Review the OPS/REXX code that EasyRule built.** Computer Associates recommends that you review the code EasyRule generated for your rule. To do so, choose option 3, for BROWSE, on the EasyRule Final Options Menu. Notice that 3 has been specified in the sample panel shown here:



```
EasyRule --------------- XE09 --- O P S V I E W --------------- Subsystem OPSS
OPTION ===> 3
        EEEEE    AAAA   SSSS   YY  YY  RRRR   UU  UU   LL      EEEEE
        EE      AA  AA  SS      YYYY   RR  R  UU  UU   LL      EE
        EEEEE   AAAAAA  SSSSS    YY    RRRR   UU  UU   LL      EEEE
        EE      AA  AA    SS    YY     RR R   UU  UU   LL      EE
        EEEEE   AA  AA  SSSS   YY      RR  R   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)
```

Lesson 5 Activity (continued)

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



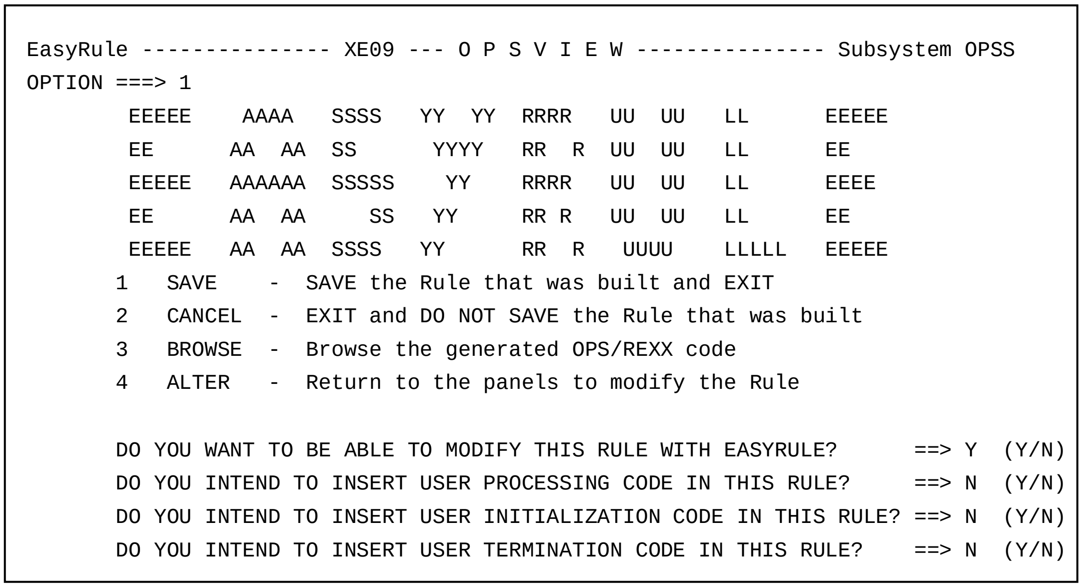
```
BROWSE -- EASY_RULE_BROWSE ------------------------- LINE 00000000 COL 001 080
COMMAND ===>                                                    SCROLL ===> PAGE
******************************* Top of Data ***********************************
)MSG  IEF287I
/**************************************************************************/
/*   Rule Name:      NOTCTLG                                             */
/*   Rule Type:      Message                                             */
/*   Rule Function:  Increase the visibility of data set catalog         */
/*                   failures as they are indicative of production       */
/*                   problems.                                           */
/*   Author:         TSOUSER                                             */
/*   Related Rules:  MNSTATUS (IEF285I)                                  */
/*   History:        99/11/17 - Original Development                     */
/**************************************************************************/
)PROC
  MSG.DESC = OPSBITS("SYSFAIL")
  Address "OPER"
    "$d m "msg.jobnm",received "msg.text
```

The panel shown above 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. This table shows the OPS/REXX code and the panel entries that correspond to it:

OPS/REXX Code	Panel Entry
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

Lesson 5 Activity (continued)

24. **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 --------------- XE09 --- O P S V I E W --------------- Subsystem OPSS
OPTION ===> 1
        EEEEE    AAAA   SSSS   YY  YY  RRRR   UU  UU   LL      EEEEE
        EE      AA  AA  SS      YYYY   RR  R  UU  UU   LL      EE
        EEEEE   AAAAAA  SSSSS    YY    RRRR   UU  UU   LL      EEEE
        EE      AA  AA     SS   YY      RR R   UU  UU   LL      EE
        EEEEE   AA  AA  SSSS   YY      RR  R   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.

Lesson 6 Activity - Testing Rules

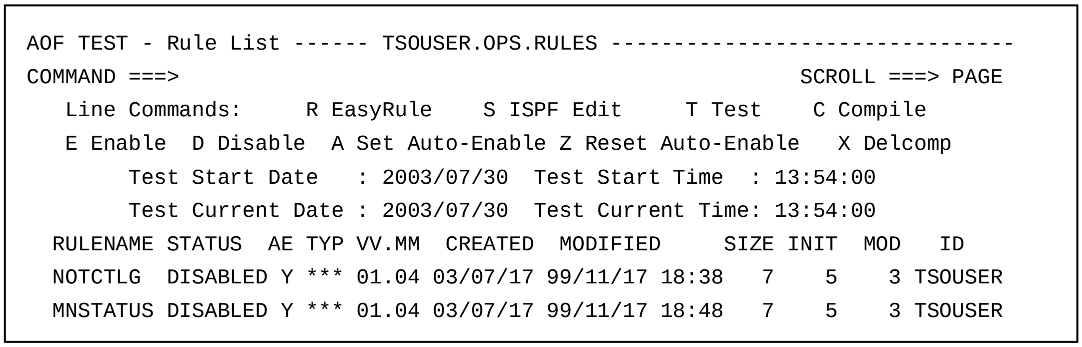
Task

- Use the OPSVIEW Editors option to test the rules named NOTCTLG and MNSTATUS, which you created in Activity 2.2.

Steps to Take

1. **Is the AOF Test Rule List panel displayed?**

   For this step, you must ensure that the AOF Test Rule List panel is being displayed, an example of which is shown here.



```
AOF TEST - Rule List ------ TSOUSER.OPS.RULES -------------------------------
COMMAND ===>                                                SCROLL ===> PAGE
   Line Commands:     R EasyRule    S ISPF Edit     T Test    C Compile
   E Enable  D Disable  A Set Auto-Enable Z Reset Auto-Enable   X Delcomp
        Test Start Date   : 2003/07/30  Test Start Time  : 13:54:00
        Test Current Date : 2003/07/30  Test Current Time: 13:54:00
  RULENAME STATUS  AE TYP VV.MM  CREATED  MODIFIED     SIZE INIT  MOD   ID
  NOTCTLG  DISABLED Y *** 01.04 03/07/17 99/11/17 18:38   7    5    3 TSOUSER
  MNSTATUS DISABLED Y *** 01.04 03/07/17 99/11/17 18:48   7    5    3 TSOUSER
```

   Action:

   If the Rule List panel is being displayed, go to “Step 2: Enable your rule and select it for testing.” 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 and select it for testing**

   This step prepares your rule for testing.

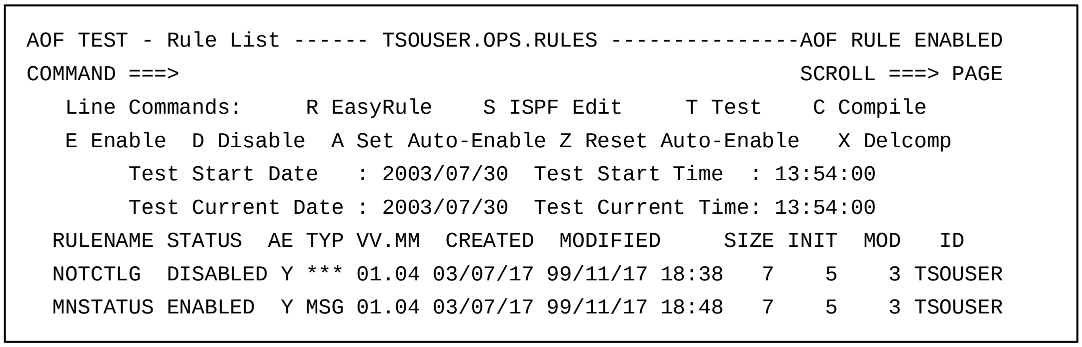
   Actions:

   - On the Rule List Panel, type E to the left of the desired rule name.

Lesson 6 Activity - Testing Rules (continued)

- Press Enter. In response, two fields on the Rule List Panel (as shown in the example screen below) change. This example shows how the panel looks after the changes. Notice that for the rule named MNSTATUS, 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: If your rule contains syntax errors, the E command fails.



```
AOF TEST - Rule List ------ TSOUSER.OPS.RULES ---------------AOF RULE ENABLED
COMMAND ===>                                                 SCROLL ===> PAGE
   Line Commands:     R EasyRule    S ISPF Edit     T Test    C Compile
   E Enable  D Disable  A Set Auto-Enable Z Reset Auto-Enable   X Delcomp
        Test Start Date   : 2003/07/30  Test Start Time  : 13:54:00
        Test Current Date : 2003/07/30  Test Current Time: 13:54:00
  RULENAME STATUS  AE TYP VV.MM  CREATED  MODIFIED     SIZE INIT  MOD   ID
  NOTCTLG  DISABLED Y *** 01.04 03/07/17 99/11/17 18:38   7    5    3 TSOUSER
  MNSTATUS ENABLED  Y MSG 01.04 03/07/17 99/11/17 18:48   7    5    3 TSOUSER
```

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

3. **Test your rule**

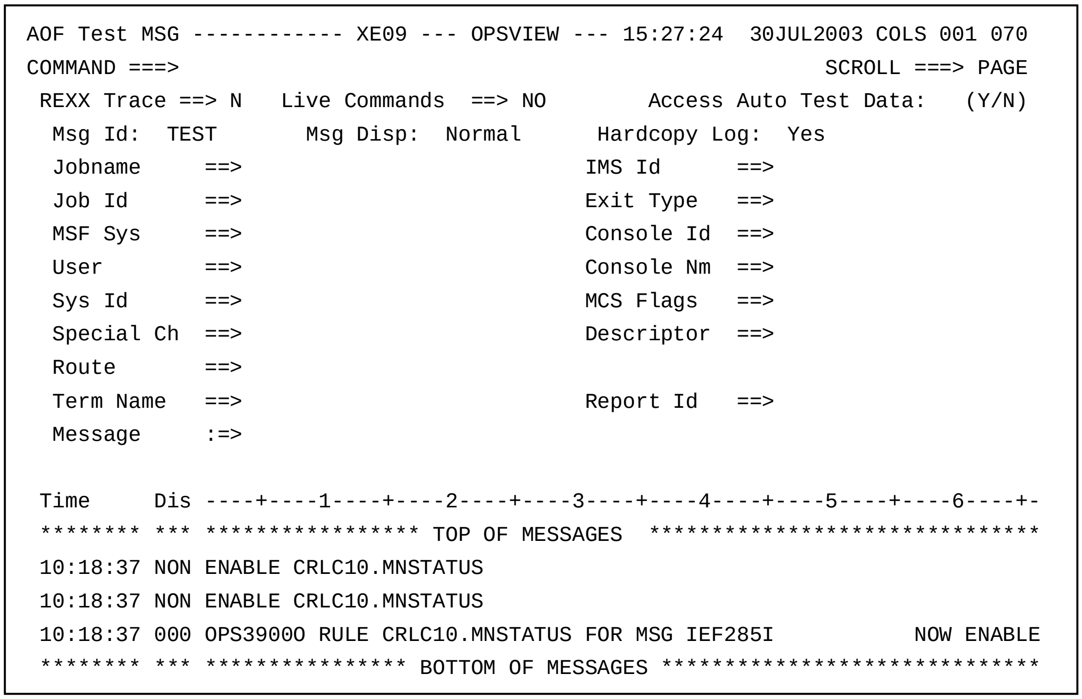
   When you select a message rule for testing, the AOF takes you to the AOF Test MSG panel. The panel prompts you for information about the message rule you want to test. A sample panel appears below. Note that at the bottom of this panel, a portion of the OPSLOG, OPS/MVS’s powerful and flexible system log, is displayed. The format of this example OPSLOG results from a Display command being entered in the Command field. This command is:

   `D TIME DISP`

   This command tells 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

Lesson 6 Activity - Testing Rules (continued)



```
AOF Test MSG ------------ XE09 --- OPSVIEW --- 15:27:24  30JUL2003 COLS 001 070
COMMAND ===>                                                  SCROLL ===> PAGE
 REXX Trace ==> N   Live Commands  ==> NO       Access Auto Test Data:   (Y/N)
  Msg Id:  TEST       Msg Disp:  Normal      Hardcopy Log:  Yes
  Jobname     ==>                          IMS Id      ==>
  Job Id      ==>                          Exit Type   ==>
  MSF Sys     ==>                          Console Id  ==>
  User        ==>                          Console Nm  ==>
  Sys Id      ==>                          MCS Flags   ==>
  Special Ch  ==>                          Descriptor  ==>
  Route       ==>
  Term Name   ==>                          Report Id   ==>
  Message     :=>

 Time     Dis ----+----1----+----2----+----3----+----4----+----5----+----6----+-
 ******** *** ***************** TOP OF MESSAGES  ******************************
 10:18:37 NON ENABLE CRLC10.MNSTATUS
 10:18:37 NON ENABLE CRLC10.MNSTATUS
 10:18:37 000 OPS3900O RULE CRLC10.MNSTATUS FOR MSG IEF285I           NOW ENABLE
 ******** *** **************** BOTTOM OF MESSAGES ******************************
```

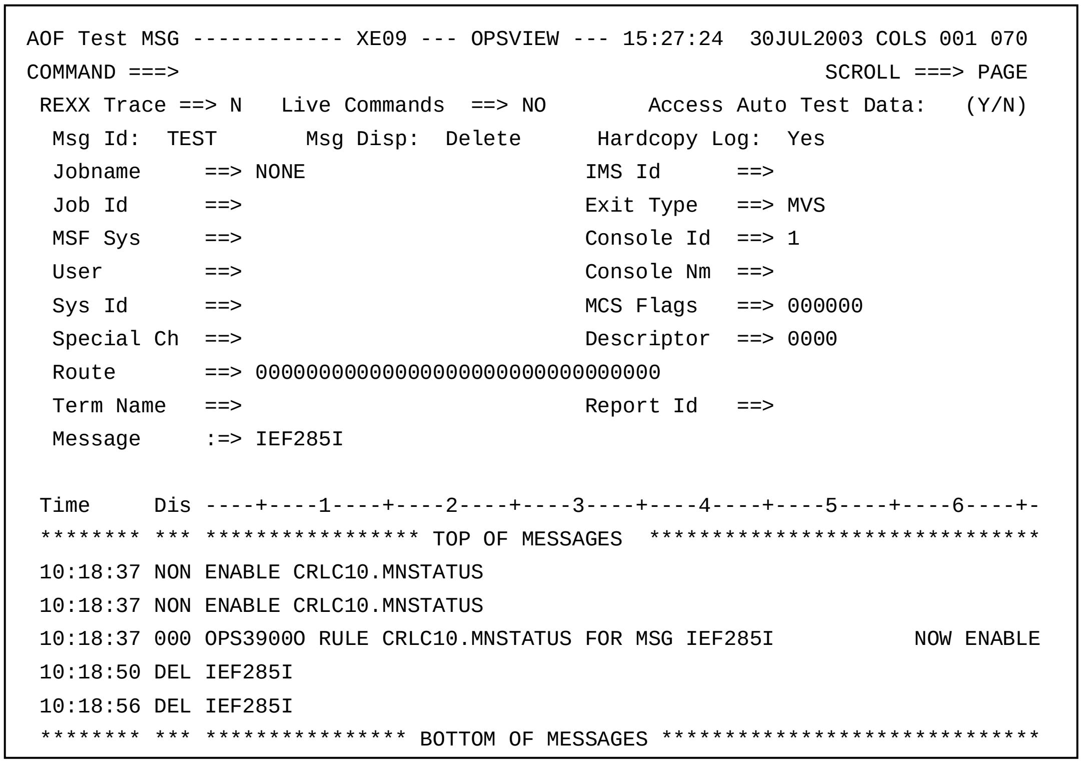
Actions: Enter values on the AOF Test MSG panel.

- Type the message ID and any text in the Message field. For example, you could type IEF285I SAMPLE MESSAGE into the Message field.
- Press Enter to run a test message against the enabled rule.

Lesson 6 Activity - Testing Rules (continued)

4. Check the test results

The sample panel that appears below shows results of your rule test.



```
AOF Test MSG ------------ XE09 --- OPSVIEW --- 15:27:24  30JUL2003 COLS 001 070
COMMAND ===>                                                  SCROLL ===> PAGE
 REXX Trace ==> N   Live Commands  ==> NO        Access Auto Test Data:   (Y/N)
  Msg Id:  TEST       Msg Disp:  Delete      Hardcopy Log:  Yes
  Jobname     ==> NONE                      IMS Id      ==>
  Job Id      ==>                           Exit Type   ==> MVS
  MSF Sys     ==>                           Console Id  ==> 1
  User        ==>                           Console Nm  ==>
  Sys Id      ==>                           MCS Flags   ==> 000000
  Special Ch  ==>                           Descriptor  ==> 0000
  Route       ==> 00000000000000000000000000000000
  Term Name   ==>                           Report Id   ==>
  Message     :=> IEF285I

 Time     Dis ----+----1----+----2----+----3----+----4----+----5----+----6----+-
 ******** *** ***************** TOP OF MESSAGES  ******************************
 10:18:37 NON ENABLE CRLC10.MNSTATUS
 10:18:37 NON ENABLE CRLC10.MNSTATUS
 10:18:37 000 OPS3900O RULE CRLC10.MNSTATUS FOR MSG IEF285I           NOW ENABLE
 10:18:50 DEL IEF285I
 10:18:56 DEL IEF285I
 ******** *** **************** BOTTOM OF MESSAGES *****************************
```

Actions:

- Examine the results of the test in the OPSLOG.
- Press your PF3 key or enter END on the command line.

Lesson 7 Assessment

1. True/False: OPS/REXX adds to standard REXX a set of extensions that automate and enhance the productivity of OS/390 operations.

   True.

2. What would you use this REXX instruction for? PARSE VAR name [template]

   To assign data to one or more variables.

3. Which REXX function returns the time in this format? /* 11:56:04 */

   SAY TIME()

4. Which REXX instruction is used to group instructions together and execute them conditionally?

   DO WHILE *expression*

5. True/False: REXX programs are built upon clauses.

   True.

6. The PUSH instruction is not available in OPS/REXX. Which alternative instruction can you use to achieve the same results?

   QUEUE.

7. What would you use this REXX instruction for? IF…THEN…ELSE

   To conditionally execute an instruction or group of instructions.

8. Which REXX operator combines two strings by appending the second string to the end of the first one?

   Concatenation.

9. Name four REXX arithmetic operators.

   Add, subtract, multiple, divide, integer divide, remainder.

10. True/False: OPS/REXX programs are called rules outside the AOF environment.

    False.

Lesson 7 Assessment (continued)

11. True/False: Standard SAA REXX I/O functions are supported in OPS/REXX.

    False.

12. What would this function return? SAY DATE('U')

    The date in USA format (mm/dd/yy).

13. True/False: In OPS/REXX, a PULL instruction to an empty external data queue results in a null line being returned.

    True.

14. Define an instruction in REXX.

    One or more clauses that describe an action to be taken.

15. Describe the difference in how standard REXX and OPS/REXX resolve external subroutines.

    In standard REXX, external subroutines are resolved only when they are called during execution. In OPS/REXX, external subroutines are resolved and bound with the main program prior to execution.

Lesson 8 Assessment

1. All variables have which characteristics?
    a. They can contain character strings of up to 256 bytes.
    b. They are not used in rules.
    c. Derived names of variables can contain up to 50 characters.
    d. Their values can change while a program is running.
    e. b and d.
    f. a, c, and d.

2. Rules can use which type of variable?
    a. Global.
    b. Static.
    c. Local.
    d. Event-related.
    e. Temporary.
    f. All of the above.

3. Global variables have which characteristics?
    a. They permit serial access.
    b. They have the form GLVTEMPx.XXXX.
    c. They have the form GLOBALx.XXXX.
    d. They allow data to be shared between different rules for events that that occur in the same address space.
    e. b and c.
    f. a, b, and c.
    g. b, c, and d.

4. Static variables have which characteristics?
    a. They maintain a fixed value across multiple executions of a single rule.
    b. They permit serial access.
    c. They are compound symbols.
    d. They are available in the)INIT and)PROC sections of a rule.
    e. a, b, and c.
    f. a and c.
    g. None of the above.

Lesson 8 Assessment (continued)

5. Event-related variables have which characteristics?
    a. They are available in the)PROC and)TERM sections of a rule.
    b. They correspond to the rule event types.
    c. They are automatically provided by the AOF engine.
    d. b only.
    e. a, b, and c.
    f. b and c.

6. Dynamic variables have which characteristics?
    a. Their value can be up to 256 bytes in length.
    b. They are simple variables or non-global compound symbols.
    c. They are created each time a rule executes.
    d. a and c.
    e. a, b, and c.
    f. b and c.
    g. None of the above.

7. Temporary variables have which characteristics?
    a. They have the form GLVTEMP.XXXX.
    b. They have the form GLVEVENT.XXXX.
    c. They allow data to be shared by different rules that are processing the same event.
    d. They are automatically deleted.
    e. They are available only in the)PROC section of a rule.
    f. a, c, and d.
    g. b, c, d, and e.

8. Local variables have which characteristics?
    a. They are available only in the)PROC section of a rule.
    b. They are unique to the address space that triggered the rule.
    c. They have the form GLVJOBID.XXXX.
    d. They allow data to be shared between different rules for events that occur in the same address space.
    e. All of the above.

Lesson 9 Assessment

Write the appropriate letter next to each description below.

h Issue system messages as WTOs and WTORs.

g Control Unicenter CA-OPS/MVS components.

l Route commands to TSO.

j Issue commands for dynamic allocation, concatenation, deconcatenation, and information retrieval functions.

k Issue operator commands from an OPS/REXX program or AOF rule.

a Send UNIX commands to servers.

n Create and maintain relational tables.

b Programmatically control rules and create dynamic AOF rules.

p Route commands to server address spaces.

m Programmatically control VTAM applications.

q Dispatch an OPS/REXX program to a specially classified long running server.

r Dispatch an OPS/REXX program to a specially classified priority server.

a. ADDRESS USS
b. ADDRESS AOF
c. ADDRESS NETMAN
d. ADDRESS RDF
e. ADDRESS REXX
f. ADDRESS SYSVIEWE
g. ADDRESS OPSCTL
h. ADDRESS WTO
i. ADDRESS MSF
j. ADDRESS OPSDYNAM
k. ADDRESS OPER
l. ADDRESS TSO
m. ADDRESS EPI
n. ADDRESS SQL
o. ADDRESS SYSVIEW
p. ADDRESS OSF
q. ADDRESS OSFTSL
r. ADDRESS OSFTSP

Lesson 10 Assessment

Write down your explanations of the following rule snippets in the space provided.

1. VOLUMES=OPSDEV('V','SYS*')

   Returns a 15-word record for each SYS* volume (UCB, volume, status, type, reserve status, etc.).

2. ISITUP=OPSTATUS('A','A','VTAMA')

   Sets the variable to 1 if it is active, 0 if it is not active.

3. RTVL=OPSVALUE('GLOBAL.VTAM.STAT','E')

   Sets the variable to I (initialized), U (uninitialized), or N (does not exist).

4. DETAILS=OPSTATUS('A','I','VTAMA')

   Returns a detailed record to the external data queue, including address space information (address space ID, status, completed steps, CPU time, etc.).

5. DEVICES=OPSDEV('U','157')

   Returns information about all online devices whose device numbers contain 157 in the first three positions.

6. SAY 'ASID='OPSINFO('ASID')

   Sets a variable to the address space ID.

7. GETWTORS=OPSTATUS('R','I','*')

   Returns all WTORs to the external data queue.

8. RTVL=OPSVALUE('GLOBAL.JOB.CNT','O')

   Sets the variable to a value of GLOBAL.JOB.CNT.

9. DEVICES=OPSDEV('D','*')

   Returns information about all online devices (DASD, TAPE, UREC, COMM, etc.).

10. SAY 'SUBSYS='OPSINFO('SUBSYS')

    Sets a variable to the z/OS subsystem name that Unicenter CA-OPS/MVS is using.

Lesson 10 Assessment (continued)

11. RTVL=OPSVALUE('GLOBAL.ACTIVE.CICS','L')
    Lists all subnodes in the external data queue.

12. SAY 'EVENTTYPE='OPSINFO('EVENTTYPE')
    Sets a variable to the type of event that the rule is processing.

13. RTVL=OPSVALUE('GLOBAL.FLAG.IMSABEND','U','S0C4')
    Updates the GLOBAL variable to a value of S0C4.

14. DEVICES=OPSDEV('D','DASD')
    Returns information about all DASD devices.

Notes: