Tivoli. software

IBM

Performance Driven Automation with OMEGAMON and System Automation for z/OS

White Paper

Paul Quigley

May 2006

Table of Contents

Introduction

White Paper : Performance Driven Automation with OMEGAMON and System Automation for z/OS

Introduction

About this Paper

IBM Tivoli System Automation for z/OS (SA z/OS) provides management and automation of zSeries resources such as Applications through monitoring and event driven automation. Customers can utilize automation policy to recover failed resources, decide how often to recover the failed resources, define service windows when it is acceptable for resources to be down, and so on.

SA z/OS provides an additional function called *monitor resources* that allows users to see status gradients between up and down. These gradients can be used to indicate a resource is *degraded*. This status is stored in the Health Status for the resource.

The OMEGAMON product suite also monitors zSeries resources such as Applications as well as zSeries metrics such as the number of outstanding WTORs, paging counts, and so on. Customers can customize OMEGAMON to notify them when a threshold has been met. This notification is called an *exception*.

SA z/OS 3.1 provides new capabilities to integrate the SA z/OS *monitor resources* with *exceptions* from the OMEGAMON *Classic* monitors for MVS, CICS, IMS, and DB2. SA z/OS will connect to the OMEGAMON monitors, retrieve exceptions as defined by the system administrator, and set the Health Status appropriately.

The integration of the new capabilities in SA z/OS 3.1 with the existing monitoring and exception reporting of the OMEGAMON monitors provides the basis for *Performance Based Automation with OMEGAMON and SA z/OS*.

The OMEGAMON *Classic* monitors will be referenced as *OMEGAMON for MVS, CICS, IMS, and DB2* within this document. You will also see *OMEGAMON monitors* used when referring to all four monitors.

This document will assist you with:

- Setting up your systems: SA z/OS, OMEGAMON, and NetView definitions.
- Managing the connections between SA z/OS and the OMEGAMON monitors. These connections are VTAM sessions.
- Defining SA z/OS monitor resources to retrieve exceptions from the OMEGAMON monitors and using the exception data to influence the Health Status of one or more resources.

- Defining more complex user automation based on exceptions from the OMEGAMON monitors.
- Additional hints and tips will be provided throughout this document.

The focus of this document is the new function in SA z/OS 3.1 to integrate SA z/OS and the OMEGAMON monitor Applications. Other enhancements for SA z/OS 3.1 will also be discussed in this document, as they relate to the integration with OMEGAMON:

- Policy Data Base (PDB) restructure into a base component (*BASE) with add-on policies (*OMEGAMON, for example) specific to a set of Applications.
- Importing a set of add-on policies to an existing PDB.
- Updating the PDB from a flat file.

The author's experiences with SA z/OS 3.1 will be shared with you.

Audience

This document is intended for use by both an SA z/OS operator and system administrator.

The system administrator will be responsible for creating and updating the policy definitions to:

- Define the connection (session) parameters between SA z/OS and the OMEGAMON monitor Applications.
- Define SA z/OS monitor resources to retrieve exceptions from the OMEGAMON monitor Applications.
- (Optional) Write custom automation to be driven by exceptions from the OMEGAMON monitor Applications.

The operator will utilize the SA z/OS operator interfaces to:

- Manage the sessions between SA z/OS and OMEGAMON.
- Manage the monitor resources.
- Manage the OMEGAMON Applications and Application Groups.
- Issue OMEGAMON commands.
- And so on.

For more information about SA z/OS monitor resources you may want to read the White Paper, *IBM Tivoli System Automation for z/OS 2.3: A Primer to Monitor Resources,* located at:

ftp://ftp.software.ibm.com/software/tivoli/education/WhitePapers/MTR_WhitePaper.pdf

Terminology Tip

This document will refer to systems using two names:

- The SA z/OS entry name. This is also known as the SA z/OS resource name. For example, MVSA and MVSB. The entry name will appear when importing policy definitions, for example.
- The actual system name. For example, TIVED1 and TIVED2. The actual system name will be used in the policy items such as when defining a relationship. The actual system name will also appear on the operator interface panels such as INGLIST and INGMOVE.

For example, the TIVED1 system is defined in the policy with an entry name of MVSA.

About the Author

Paul Quigley is a member of the Tivoli Technical Training Course Development organization with primary responsibilities for developing the SA z/OS customer courses. He has worked on mainframe-based systems and network management, monitoring, and automation for 25 years with an extensive background using the Tivoli NetView for z/OS platform and products. Paul has held various positions within IBM ranging from software test to product design and development to product support and has worked with many customers worldwide to understand their requirements and solve their problems.

White Paper: Performance Driven Automation with OMEGAMON and System Automation for z/OS

1 Overview

Exception data from the OMEGAMON monitors can be used to trigger automated actions in SA z/OS. The exception data can be used to influence the SA z/OS Health Status of resources and provide the basis for more complex automation.

This document will show you how to use the SA z/OS Customization Dialog and operator interface to:

- Define and manage the connections between the SA z/OS Automation Agent and the OMEGAMON monitors.
- Map an OMEGAMON exception to a change in the SA z/OS Health Status.
- Define automation to be driven when an exception is detected.
- Issue OMEGAMON commands from the SA z/OS agent NetView.

Any discussion involving SA z/OS Health Status must begin with an understanding of SA z/OS *monitor resources*. This document will begin with a brief review of monitor resources followed by detailed discussion of the enhancements to SA z/OS 3.1 to support integration with the OMEGAMON monitors.

This document will use several exceptions such as XREP, WAIT, FXFR, and SWPC in examples. For more information about exceptions, in general, read the *OMEGAMON for MVS Reference Manual*.

1.1 Overview of SA z/OS Monitor Resources

Monitor resources are policy objects that allow you to obtain the health state of other resources, typically Applications or Application Groups. The health state is useful when you need to know how well the resource is performing and not just that it is active.

Monitor resources use a resource type of MTR and a sixth status for the SA z/OS Automation Manager, the Health Status. The Health Status can be used to provide Application-specific performance and health monitoring information. For example, an Application may be active but it is failing to meet performance objectives defined by the system administrator. The Health Status can be set such that the Compound Status, known by the Automation Manager, is *degraded*.

There are several types of monitor resources. A **passive monitor** is one which waits for events to occur that drive automation to set the Health Status of a resource. An **active monitor** is one which proactively determines the Health Status of a resource by issuing a command called the *monitor routine.* The monitor routine performs the proactive monitoring by issuing one or more commands. In some cases you may need to combine monitor types, using an active monitor to proactively determine the Health Status of a resource as well as a passive monitor with automation defined to set the Health Status of a resource.

The monitor routine is typically a REXX routine written by the customer and is specific to the resource or type of resource being monitored. In general, the monitor routine will issue one or more commands to gather data about the resource and then set a return code based on the data collected. The return code is then used by SA z/OS to indicate the health of the resource.

The possible values for the Health Status (with the monitor routine return code shown in parentheses) are:

- **NORMAL (3)**: The health of a resource is good.
- **WARNING (4)**: The health of a resource is becoming degraded.
- **MINOR (5)**: Similar to the warning status, but more severe.
- **CRITICAL (6)**: Similar to the minor status, but more severe.
- **FATAL (7)**: Similar to the critical status, but more severe. This will cause the Automation Manager Compound Status to change to PROBLEM which may, in turn, trigger an action from the manager such as moving an Application to another system.
- **DEFER (8)**: The Health Status is not set by the monitor routine. Instead, it will be set by automation or by the MESSAGES/USER DATA policy item for a monitor resource.
- **BROKEN (1)**: The monitor failed and is not capable of running. No further Health Status is provided.
- **FAILED (2)**: The monitor failed and will be rescheduled. No Health Status is provided.
- **UNKNOWN**: The Health Status has not been determined. The monitor is not running. It may have been stopped or has not been started.

BROKEN, FAILED, and UNKNOWN are set by the SA z/OS processes based on the status of the monitor itself. NORMAL, WARNING, MINOR, CRITICAL, and FATAL are set by the monitor routine. DEFER is intended for automation of OMEGAMON exceptions.

SA z/OS 3.1 provides a monitor routine (INGMTRAP) for OMEGAMON exception data so it is essential that you have a basic understanding of SA z/OS monitor resources. INGMTRAP will be discussed in more details later. You may also choose to write your own monitor routine for OMEGAMON exceptions. If so, you will need to be familiar with these return codes and their effect on the Health Status of a resource.

1.2 Overview of SA z/OS Support for OMEGAMON

The SA z/OS 3.1 support for OMEGAMON can be grouped into two main functional areas.

The first functional grouping is the ability to monitor and manage the OMEGAMON Applications and Application Groups. SA z/OS ships a set of default policy definitions for the OMEGAMON Applications and Application Groups. This document will show you how to import the default policy and customize it so that you can manage the OMEGAMON Applications and Application Groups in your environment.

The second functional grouping is the ability to influence the SA z/OS Health Status for resources based on exception data from the OMEGAMON monitors. SA z/OS uses monitor resources to retrieve the exception data from the OMEGAMON monitors. The exception data is retrieved using a specialized monitor routine (INGMTRAP) provided by SA z/OS.

Each OMEGAMON monitor can generate an exception when a threshold has been reached. The thresholds are defined within the OMEGAMON monitor itself and should not be confused with the recovery thresholds provided by SA z/OS.

SA z/OS requires a session between the Automation Agent and each OMEGAMON monitor to retrieve the exception data. This document will show you how to define and manage the sessions.

You will see how to map an OMEGAMON exception to a change in SA z/OS Health Status plus several techniques for more complex automation of the OMEGAMON exceptions.

An additional benefit of the OMEGAMON integration provides support for operators to issue OMEGAMON commands using the sessions between the agent NetView and the OMEGAMON monitors. You will see examples of this later in this document.

2 Manage the OMEGAMON Resources

The OMEGAMON Applications and Application Groups can be managed and automated by SA z/OS in the same manner as most any other z/OS Application or Basic Application Group. SA z/OS provides a default set of policy definitions that you can import to get started. Once you have the policy customized you can use existing SA z/OS commands such as INGLIST to manage the resources.

2.1 Import the OMEGAMON Add-on Policy

The first step in the process of managing the OMEGAMON resources is to define the policy for each Application and Application Group. The simplest method is to import the policy definitions from the sample *OMEGAMON add-on policy. After the policies are imported you can then customize them for your particular environment.

This document groups the required tasks into eight steps to the import and the customization the OMEGAMON add-on policy. The Data Management functions of the Customization Dialog are used to import the add-on policy.

Open your Policy Data Base (PDB). This document will use a PDB called WP_PDB. It has already been created from the *BASE PDB and contains a set of customized definitions for our environment.

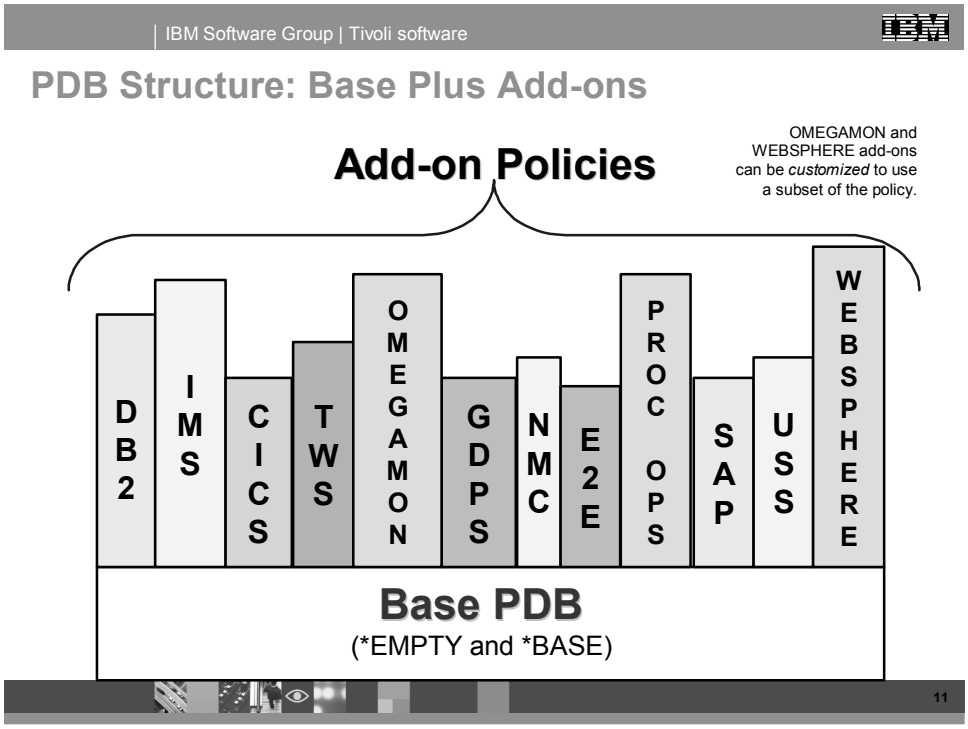
2.1.1 Data Management Panel

*2.1.1.1 Step 1: Import the *OMEGAMON Add-on Policy*

To manage a resource requires one or more policy definitions. You can create the policy data manually or you can use the samples as a starting point. Creating the policy definitions manually can be labor intensive, requiring that you define many policy items. This process can also yield typing errors. Importing a sample set of policy reduces the time it takes to implementing automation for the OMEGAMON Applications.

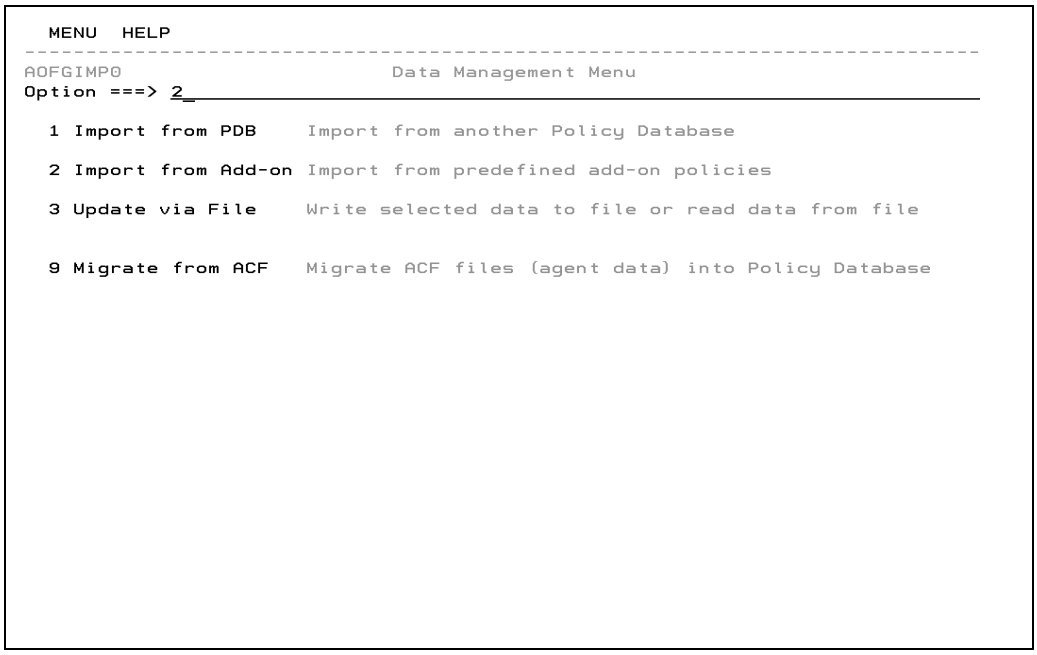
This document will show you how to use the sample policy definitions. The first step is to import the *OMEGAMON add-on policy.

Add-on policy is a new term and applies to sets of policy towers unique to each set of Applications. This slide shows the new structure of the PDB for SA z/OS 3.1. There is a base set of policy (*BASE) and many add-on policies:



Assuming you already have a PDB created such as the WP_PDB used in this document you can begin with the import of the sample OMEGAMON policies in the *OMEGAMON add-on.

Using the *Data Management Menu* panel (AOFGIMP0), select option **2** to import from an add-on set of policy.



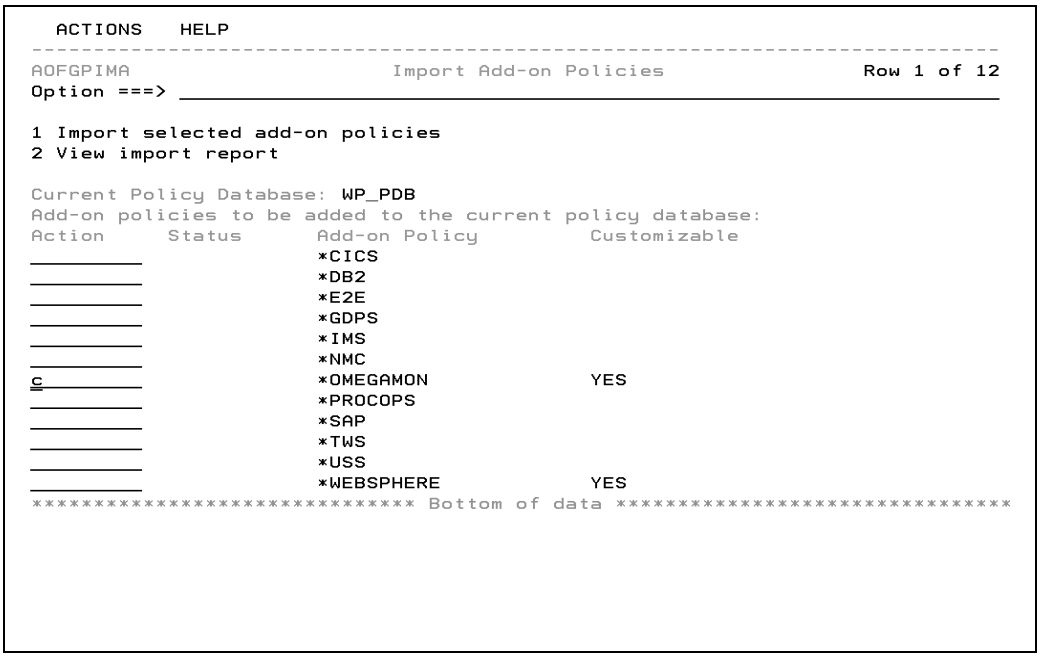
Option two will display the *Import Add-on Policies* panel (AOFGPIMA) with a list of the available add-on policies.

2.1.1.2 Step 2: Customize the *OMEGAMON Add-on

The *Import Add-on Policies* panel is used to select the add-on policy to be imported. You can select as many add-on policies from this panel as you need.

Two of the add-on policies (*OMEGAMON and *WEBSPHERE) can be *customized* to import a subset of their policy definitions. The *OMEGAMON add-on will be customized in this example to select only the OMEGAMON II policies.

Enter a **C** next to the *OMEGAMON add-on and press **Enter** to *customize* it.

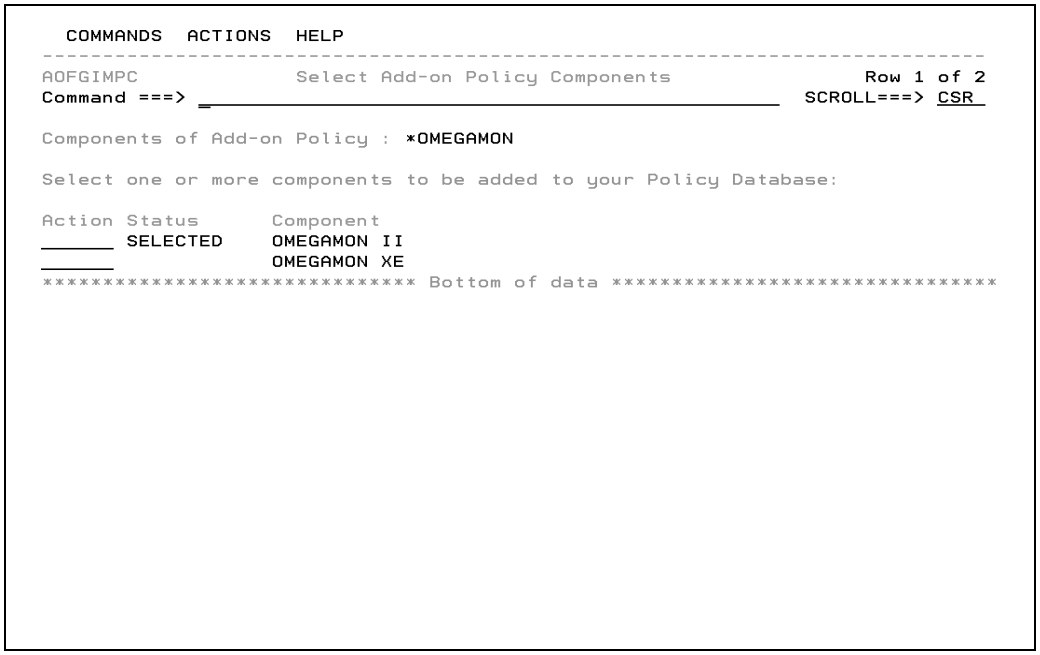


2.1.1.3 Step 3: Remove the OMEGAMON for XE Policies

When the *Select Add-on Policy Components* panel (AOFGIMPC) is displayed, the OMEGAMON II and OMEGAMON XE policies are SELCTED.

Remove the OMEGAMON XE policies with an *Action* of **M**.

The OMEGAMON II policies should still have a status of SELECTED.



This will import all of the policies for the OMEGAMON II resources, including OMEGAMON II for MVS, CICS, IMS, and DB2. You will further refine the list of policies to import in the next step.

Press **PF3** to return to the *Import Add-on Policies* panel.

2.1.1.4 Step 4: Import the Add-on Policies

With the *OMEGAMON add-on CUSTOMIZED, select option **1** (Import selected add-on policies), shown below, to work with all of the policies for the OMEGAMON II resources.

```
  ACTIONS   HELP
------------------------------------------------------------------------------
AOFGPIMA                    Import Add-on Policies              Row 1 of 12
Option ===> 1_

1 Import selected add-on policies
2 View import report

Current Policy Database: WP_PDB
Add-on policies to be added to the current policy database:
Action     Status      Add-on Policy         Customizable
_________              *CICS
_________              *DB2
_________              *E2E
_________              *GDPS
_________              *IMS
_________              *NMC
_________  CUSTOMIZED  *OMEGAMON            YES
_________              *PROCOPS
_________              *SAP
_________              *TWS
_________              *USS
_________              *WEBSPHERE            YES
******************************* Bottom of data ********************************
```

The *Entries of selected Add-on Policies* panel (AOFGPIM4) will be displayed next with the complete set of policies for the OMEGAMON II resources.


```
  COMMANDS  ACTIONS  HELP
------------------------------------------------------------------------------
AOFGPIM4          Entries of selected Add-on Policies              Row 1 of 26
Command ===> _                                                SCROLL===> CSR

Action Entry Name            Type C D Short Description
______ SYSPLEX1              GRP    Y Placeholder. Original in *BASE
______ SYS1                  SYS      Placeholder. Original defined in *BASE
______ SYS2                  SYS      Placeholder. Original defined in *BASE
______ SYS3                  SYS      Placeholder. Original defined in *BASE
______ OMII_CICS             APG      OMII CICS Application Group
______ OMII_DB2              APG      OMII DB2 Application Group
______ OMII_GROUP            APG      APG used to link APLs to SYSs
______ OMII_IMS              APG      OMII IMS Application Group
______ OMII_MVS              APG      OMII MVS Application Group
______ C_OM                  APL  *   OMEGAMON Application Class
______ OMIICSUB              APL      Candle subsystem
______ OMIIC20               APL      OMII CICS CUA interface
______ OMIID2                APL      OMII DB2 CUA interface
______ OMIIETE               APL      OMII MVS End-To-End response monitor
______ OMIII2                APL      OMII IMS CUA interface
______ OMIIM2                APL      OMII MVS CUA interface
______ OMIIM2CS              APL      OMII MVS CSA analyser
______ OMIIM2EZ              APL      OMII MVS Epilog zoom
______ OMIIM2HD              APL      OMII MVS Historical data interface
______ OMIIM2HI              APL      OMII MVS Epilog Collector
______ OMIIM2RC              APL      OMII MVS Realtime Collector
______ OMIIOC0               APL      OMII CICS STC
______ OMIIOI0               APL      OMII IMS STC
______ OMIIO2                APL      OMII DB2 STC
______ OMVIEW                APL      OMEGAVIEW
______ SESS_AUTOOPS          AOP      OMEGAMON Session Operators
```

Notice the policy definitions include a sysplex group, three systems, several Application Groups, automation operators, and many Application definitions for all of the OMEGAMON II Applications (MVS, CICS, IMS, DB2, OMEGAVIEW, and so on).

2.1.1.5 Step 5: Remove Unwanted Applications

Unless you are running all of the OMEGAMON II Applications you will need to remove several entries from this list. Start with the sysplex group and the SYS3 system. This document uses two system entries, MVSA and MVSB. The third system entry definition, SYS3, is not needed. The sysplex group definition is already included in the WP_PDB being used (the D column indicates this is a duplicate entry).

Note: In general, you should rename the systems (SYS1 and SYS2) at this point. If you do not then you will have to edit each Application and Application group and link them to the systems. Renaming the systems in this step will ensure the OMEGAMON resources are automatically linked to their appropriate systems.

There are two methods to rename the systems:

- Tab to the *Action* column and enter an **R** to rename the entry. You will see a pop-up window with a field where you can type the new name.
- Tab to the *Entry Name* column and type the new name in place of the existing name.

This document focuses on the OMEGAMON II for MVS Applications only, so all others will be removed from the list as well. Tab to each entry and enter an **M** to remove it from the list. You can use the text supplied in the *Short Description* column to help you determine which entries to remove or keep.

When you have finished customizing the list of entries to import, your panel should look similar to:

```
  COMMANDS  ACTIONS  HELP
------------------------------------------------------------------------------
AOFGPIM4          Entries of selected Add-on Policies              Row 1 of 14
Command ===> _                                                SCROLL===> CSR

Action Entry Name           Type C D Short Description
______ MVSA                 SYS    Y Placeholder. Original defined in *BASE
______ MVSB                 SYS    Y Placeholder. Original defined in *BASE
______ OMII_GROUP           APG      APG used to link APLs to SYSs
______ OMII_MVS             APG      OMII MVS Application Group
______ C_OM                 APL  *   OMEGAMON Application Class
______ OMIICSUB             APL      Candle subsystem
______ OMIIETE              APL      OMII MVS End-To-End response monitor
______ OMIIM2               APL      OMII MVS CUA interface
______ OMIIM2CS             APL      OMII MVS CSA analyser
______ OMIIM2EZ             APL      OMII MVS Epilog zoom
______ OMIIM2HD             APL      OMII MVS Historical data interface
______ OMIIM2HI             APL      OMII MVS Epilog Collector
______ OMIIM2RC             APL      OMII MVS Realtime Collector
______ SESS_AUTOOPS         AOP      OMEGAMON Session Operators
****************************** Bottom of data ********************************
```

The result will be a list of policy definitions for eight Applications, one Application class, two Application Groups, and the SESS_AUTOOPS AOP policy. These are the entries that will be imported to the WP_PDB.

Notice that there are two system entries defined: MVSA and MVSB. They were created by renaming SYS1 and SYS2.

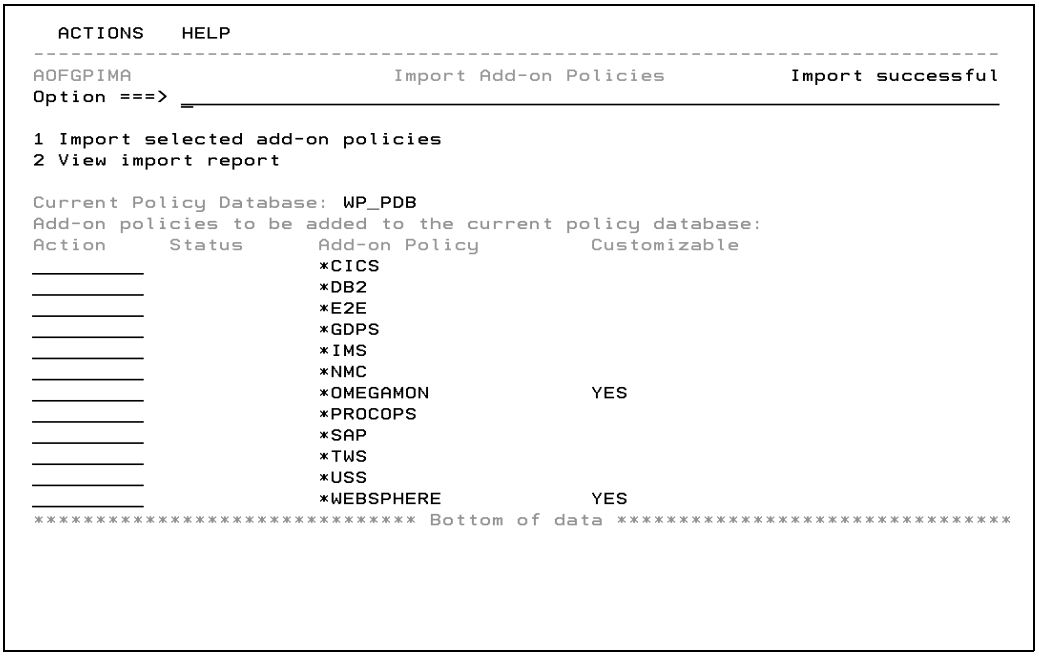
Note: If you look closely at the previous panel you will see that SYS1and SYS2 were not identified as duplicate entries. As soon as they were renamed SA z/OS identified them as duplicates. By renaming the system entries in this step the links are updated, saving you time later.

Application classes will be identified by an asterisk (*) in the *C* (class) column. In this example there is one Application class, C_OM.

Entries that already exist will be identified by a **Y** in the *D* (duplicate) column. Duplicate entries are not copied during the import process. New links and the objects the links point to will be copied. In this example both MVSA and MVSB are identified as duplicate entries.

This document retains the sample names such as OMIIM2RC and later renames the jobname and subsystem name to CANSM2RC. That will cause CANSM2RC to be used when displayed on the SA z/OS panels. If that is confusing you can rename OMII* Applications during this step to match the subsystem name.

Press **Enter** and the process to import the policies will begin. When the import completes you should see the *Import Add-on Policies* panel with the message, *Import successful*, in the upper right corner of the panel:



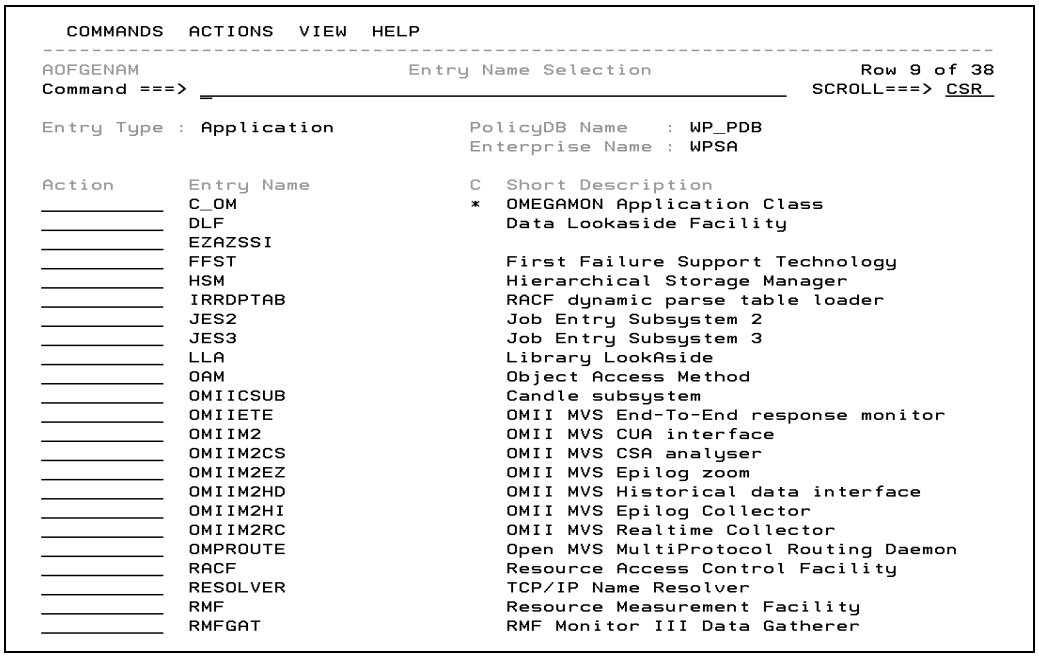
The import of the sample OMEGAMON II policy subset is complete. The WP_PDB now contains sample OMEGAMON II for MVS policy definitions.

The next section of this document discusses how to modify the imported policy definitions to change policy items such as the job name of each OMEGAMON Application, for example.

2.2 Customize the OMEGAMON Policy: An Example

Open the WP_PDB and select Applications (option **6**, APL) to see all defined Applications, including the OMEGAMON II for MVS Applications that were imported.

You should see an *Entry Name Selection* panel (AOFGENAM) similar to:



Notice the OMII* Applications. These are the OMEGAMON II for MVS Applications that were created during the import process.

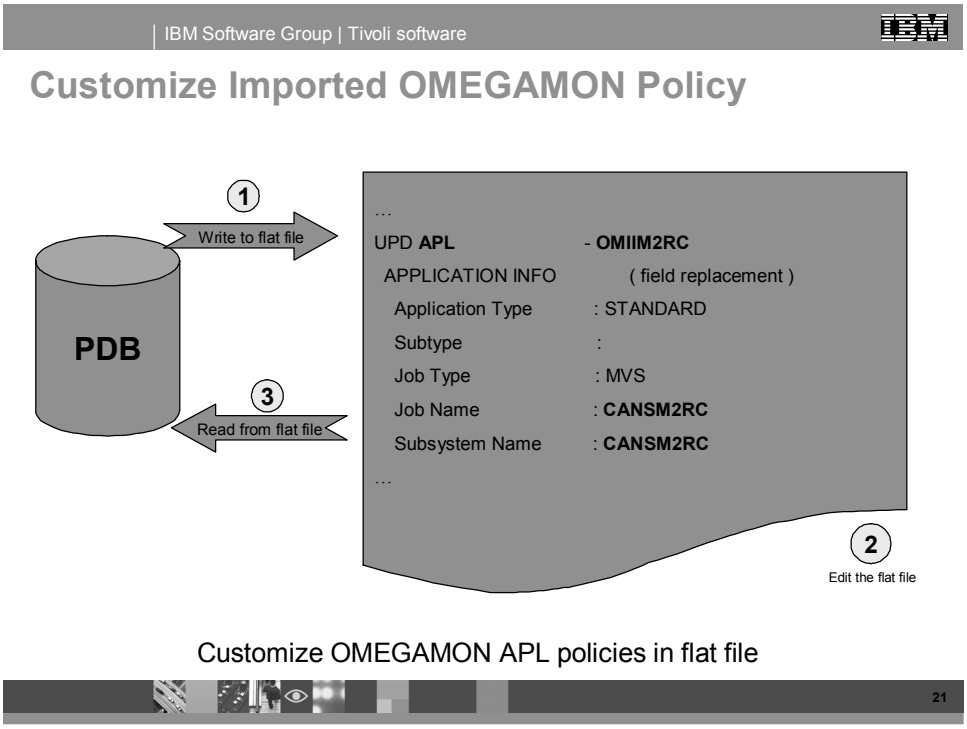
2.2.1 Step 6: Customize the Imported Application Policies for Your Environment

Now that you have imported the policy definitions you will need to edit them to change items such as the job name and subsystem name to match your system environment.

You can modify the imported policies by:

- Using the Customization Dialog panels to modify the policy manually for each OMII* Application.
- Using another new function in SA z/OS 3.1 to write the policy definitions to a flat file, edit the flat file, and then read the modified policy back in to the PDB.

This document will show you how to use the flat file update feature in SA z/OS 3.1.



This slide provides an overview of the flat file update process. Select option **3** (Update via File) of the *Data Management* panel to use the flat file update process. The process has three basic steps:

1. In the first step you select the Applications and policy items that you want to write to the flat file.

   In this example, select all OMII* Applications and write their APPLICATION INFO policy item to a flat file. You can name the flat file (for example, ING310.OMEGAMON.APLS) or you can let SA z/OS name it by appending **.UPD** to the data set name.

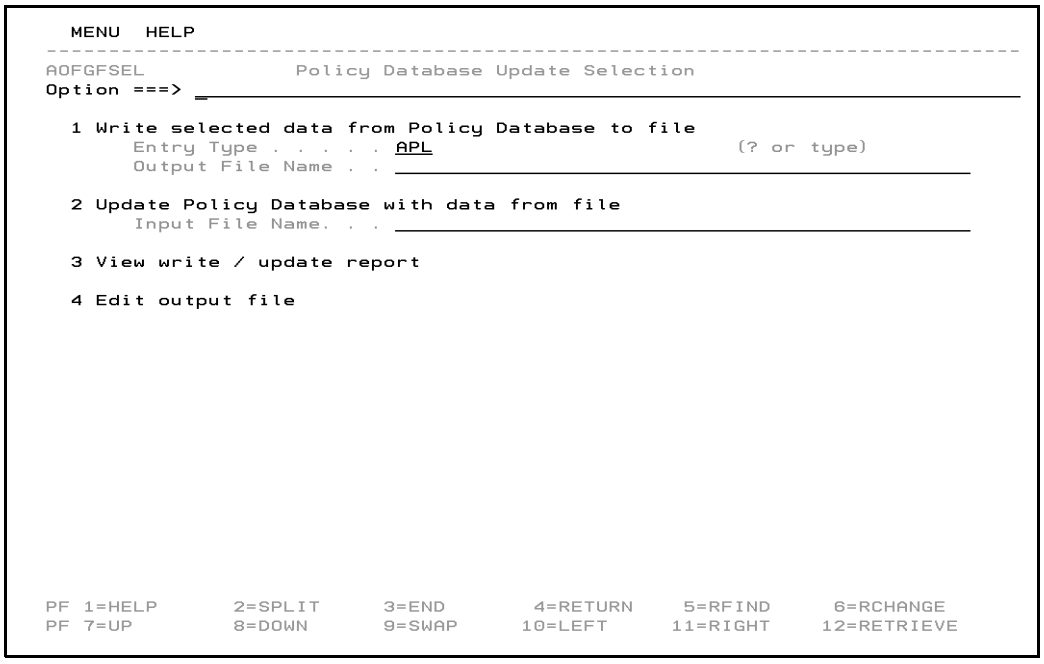
2. In the second step you edit the policy in the flat file. Entries that you may need to change can include the job name and subsystem name fields.

   In this example, the OMIIM2RC Application job name must be changed to CANSM2RC and the subsystem name must be changed to CANSM2RC.

   Make similar modifications to the other OMII* Application policies in the flat file.

3. After you have saved your changes, the flat file can be read back in to the PDB by using the flat file update function. You should see the message, *Data written to PDB*, if the flat file update is successful.

This is an example of the *Policy Database Update Selection* panel, AOFGFSEL:



Use this panel to perform the flat file updates. This is a very brief summary of the function provided with each option:

- Option one: Write the Application policy to the flat file. If you do not enter a name in the *Output File Name* field then SA z/OS will use the PDB name appended with .UPD.
- Option two: Write the data from the flat file in to the PDB.
- Option three: Display the report file for both write and read actions.
- Option four: Can be used to edit the contents of the flat file.

2.2.2 Step 7: Link Application Groups

This step is not required if you renamed your system entries (from SYS1 and SYS2 to MVSA and MVSB, for example) in step 5.

The OMEGAMON II for MVS Applications are automatically linked to two Application Groups, OMII_MVS and OMII_GROUP. If the Application Groups were not linked to a system you will need to link them in this step.

Select the WHERE USED policy item for the OMII_MVS and OMII_Group Application Groups and select the systems for each group. The systems used in this document are named MVSA and MVSB.

2.2.3 Step 8: Build the Automation Configuration Files (ACFs)

At this point, you have imported policies for the OMEGAMON II for MVS Applications and Application Groups, linked them to two system entries (MVSA and MVSB), and customized the policy definitions for those systems.

It is now time to build your ACFs from the WP_PDB.

2.3 Manage the OMEGAMON Applications

The OMEGAMON Applications and Application Groups can be managed with the INGLIST command. The OMEGAMON Applications and Application Groups will appear in the INGLIST display like any other SA z/OS resource.

2.3.1 INGLIST Example

For example, to view the OMEGAMON Application Groups, enter **INGLIST OM***.

```
INGKYST0                  SA z/OS  - Command Dialogs     Line  1     of 2
Domain ID   = AOFDA       -------- INGLIST   ---------    Date = 01/31/06
Operator ID = NETOP1          Sysplex = SYSPLEX1          Time = 16:12:44
CMD: A Update    B Start     C Stop      D INGRELS   E INGVOTE   F INGINFO
     G Members   H DISPTRG   I INGSCHED  J INGGROUP  M DISPMTR   / scroll
CMD Name          Type System    Compound      Desired      Observed    Nature
--- ------------ ---- --------  ------------  -----------  ----------  --------
 _  OMII_MVS     APG  TIVED1    SATISFACTORY  AVAILABLE    AVAILABLE   BASIC
 _  OMII_MVS     APG  TIVED2    SATISFACTORY  AVAILABLE    AVAILABLE   BASIC

Command ===> _
 PF1=Help    PF2=End      PF3=Return   PF4=DISPSTAT  PF5=Filters   PF6=Roll
                          PF9=Refresh PF10=Previous PF11=Next     PF12=Retrieve
```

This is the INGLIST panel with the OMII_MVS Application Group for two systems. OMII_MVS is a Basic System Application Group. To see the group members, enter **G** for the CMD:

```
INGKYST0                  SA z/OS  - Command Dialogs     Line  1     of 7
Domain ID   = AOFDA       -------- INGLIST   ---------    Date = 01/31/06
Operator ID = NETOP1          Sysplex = SYSPLEX1          Time = 16:17:13
CMD: A Update    B Start     C Stop      D INGRELS   E INGVOTE   F INGINFO
     G Members   H DISPTRG   I INGSCHED  J INGGROUP  M DISPMTR   / scroll
CMD Name          Type System    Compound      Desired      Observed    Nature
--- ------------ ---- --------  ------------  -----------  ----------  --------
 _  CANSETE       APL  TIVED2    SATISFACTORY  AVAILABLE    AVAILABLE
 _  CANSM2        APL  TIVED2    SATISFACTORY  AVAILABLE    AVAILABLE
 _  CANSM2CS      APL  TIVED2    SATISFACTORY  AVAILABLE    AVAILABLE
 _  CANSM2EZ      APL  TIVED2    SATISFACTORY  AVAILABLE    AVAILABLE
 _  CANSM2HD      APL  TIVED2    SATISFACTORY  AVAILABLE    AVAILABLE
 _  CANSM2HI      APL  TIVED2    SATISFACTORY  AVAILABLE    AVAILABLE
 _  CANSM2RC      APL  TIVED2    SATISFACTORY  AVAILABLE    AVAILABLE

Command ===> _
 PF1=Help    PF2=End      PF3=Return   PF4=DISPSTAT  PF5=Filters   PF6=Roll
                          PF9=Refresh PF10=Previous PF11=Next     PF12=Retrieve
```

This INGLIST example shows the members of the OMII_MVS group. The members were the Applications that were imported from the *OMEGAMON add-on.

Since there are no unique functions for the managing OMEGAMON resources you can use other SA z/OS functions such as INGVOTE, INGSCHED, INGGROUP, and so on.

3 Using OMEGAMON Exceptions to Set SA z/OS Health Status

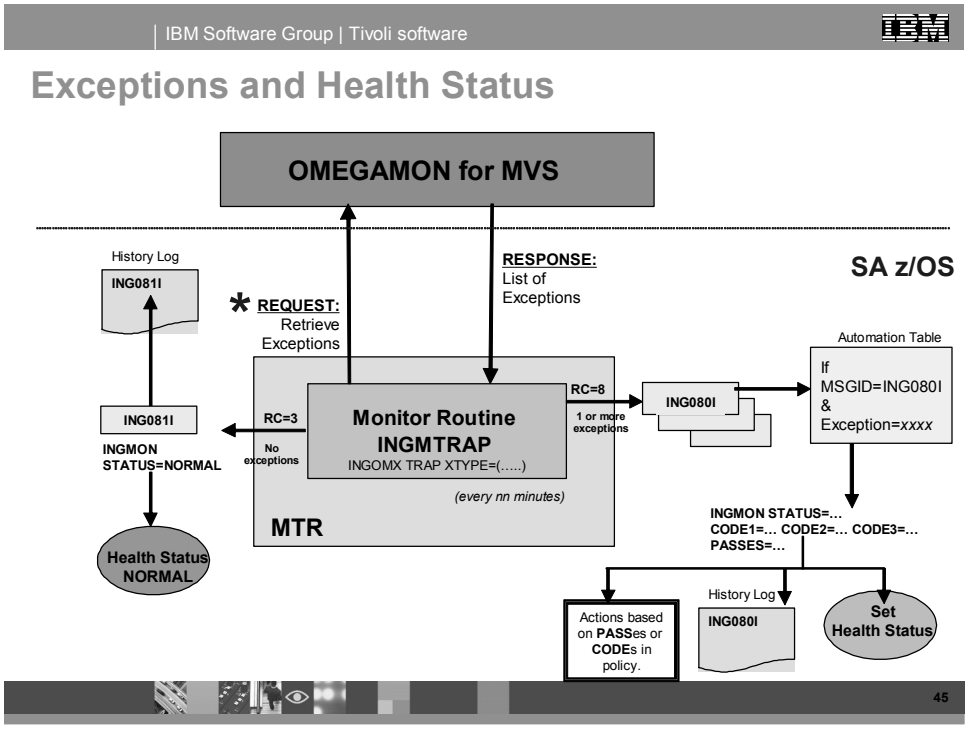
Traditionally, OMEGAMON is monitoring its resources and setting exceptions based on the thresholds defined to OMEGAMON and SA z/OS is monitoring the resources defined in the PDB.

With SA z/OS 3.1 you can integrate the data from the OMEGAMON monitors in to SA z/OS, using SA z/OS monitor resources.

The slide below illustrates SA z/OS with a monitor resource defined. The monitor routine is INGMTRAP. INGMTRAP will be scheduled on a timer basis using the monitor interval defined in the monitor resource (MTR) policy.

INGMTRAP will issue commands to retrieve one or more exceptions from an OMEGAMON monitor, OMEGAMON for MVS in this example. The commands and responses flow across a session between SA z/OS and the OMEGAMON monitor.

This chapter will show you how to define the sessions and the monitor resource definitions to set the SA z/OS Health Status.



The process begins with the INGMTRAP monitor routine requesting one or more exceptions from the OMEGAMON for MVS monitor Application.

When one or more exceptions are detected: (right side of slide)

- The monitor routine (INGMTRAP) ends with a return code of 8 (Health Status is DEFER).
- An ING080I message is generated for each exception. For example:

  ```
  ING080I XREPMON/MTR/TIVED2 OMIIMVSB OMIIMVS XREP Number of
  Outstanding Replies = 4
  ```

  Where:

  – **XREPMON/MTR/TIVED2** is the monitor resource name
  – **OMIIMVSB** is name of the session used
  – **OMIIMVS** identifies the monitor type as OMEGAMON II for MVS
  – **XREP Number of Outstanding Replies = 4** is the exception message from the OMEGAMON for MVS monitor

- Each ING080I will drive the Automation Table (AT) and, based on the policy definitions, issue one or more commands to provide automation for the exception and to set the Health Status for the monitor resource.

Note: This part of the process has been simplified to simply set the Health Status of the resource. You will see later how to process CODEs and PASSes to perform more complex automation and to set the Health Status.

If no exception is detected: (left side of slide)

- The monitor routine ends with a return code of 3 (Health Status is NORMAL).
- An ING081I message is generated. For example:

  ```
  ING081I XREPMON/MTR/TIVED2 OMIIMVSB OMIIMVS NO EXCEPTION
  FOUND
  ```

- The monitor resource Health Status is set to NORMAL.

The monitor routine, INGMTRAP, is then rescheduled based on the time interval defined for the monitor resource.

You will see how all of these pieces fit together and the policy definitions that are required. The first step is to define the connections between the SA z/OS Automation Agent and the OMEGAMON monitors. There are two new Network (NTW, option **39**) policy items in the Customization Dialog. After that you will see how to define a monitor resource to retrieve an OMEGAMON exception and set the SA z/OS Health Status.

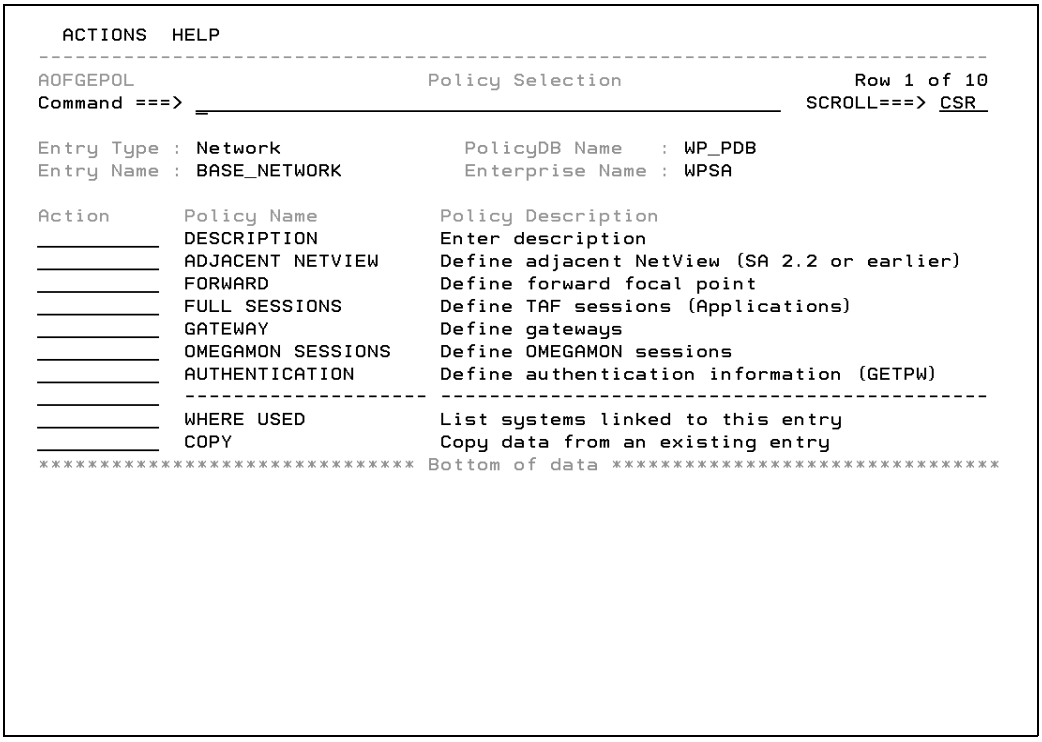
This chapter will discuss:

- Defining OMEGAMON sessions with the NTW policy.
- Defining monitor resources (MTR) policy to:

  - Retrieve OMEGAMON exceptions.
  - Influence the SA z/OS Health Status.
  - Issue a command when an exception is detected.
- Defining automation operator (AOP) policy.
- Defining relationships between the monitor resources and the OMEGAMON monitors.

3.1 Network (NTW) Policy Definitions

This is the *Policy Selection* panel (AOFGEPOL) with the list of NTW policy items.



The OMEGAMON SESSIONS and AUTHENTICATION policy items are new for this release.

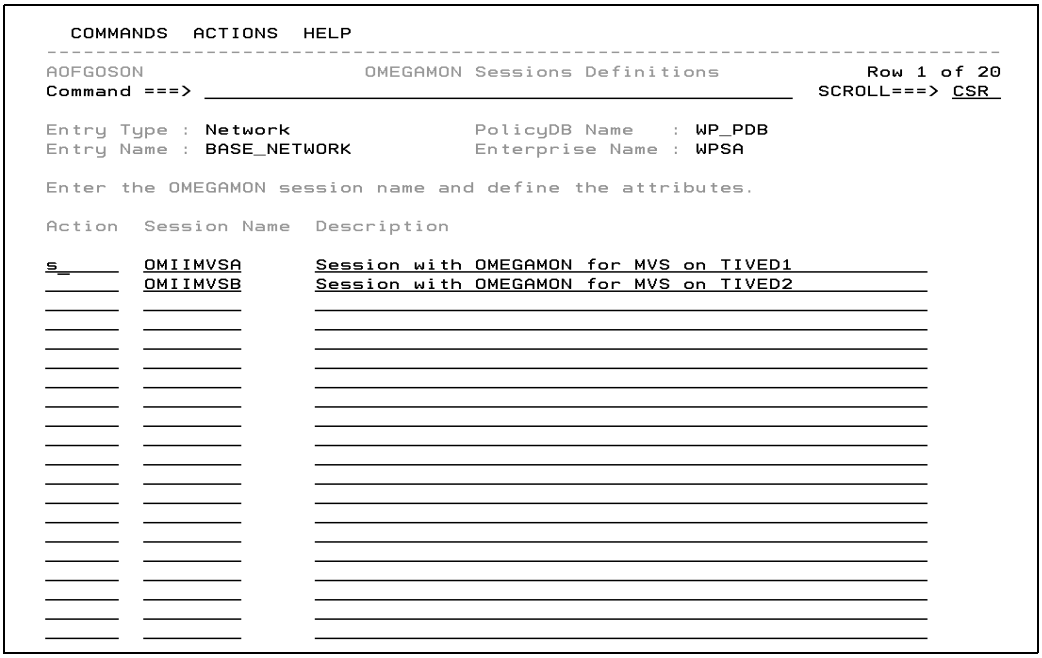
- OMEGAMON SESSIONS: Defines the connection parameters between the SA z/OS agent and an OMEGAMON monitor. The connection is a TAF (terminal access facility) full-screen session between the agent and the OMEGAMON monitor.
- AUTHENTICATION: (Optional) Defines password encryption for the connection between the agent and the OMEGAMON monitor. This policy item will be discussed in more detail when security is discussed.

3.1.1 OMEGAMON SESSIONS Policy

The OMEGAMON SESSIONS policy item defines the parameters for the TAF full-screen session between the agent and each OMEGAMON monitor. The OMEGAMON monitors can run on the same system as the agent or in another system.

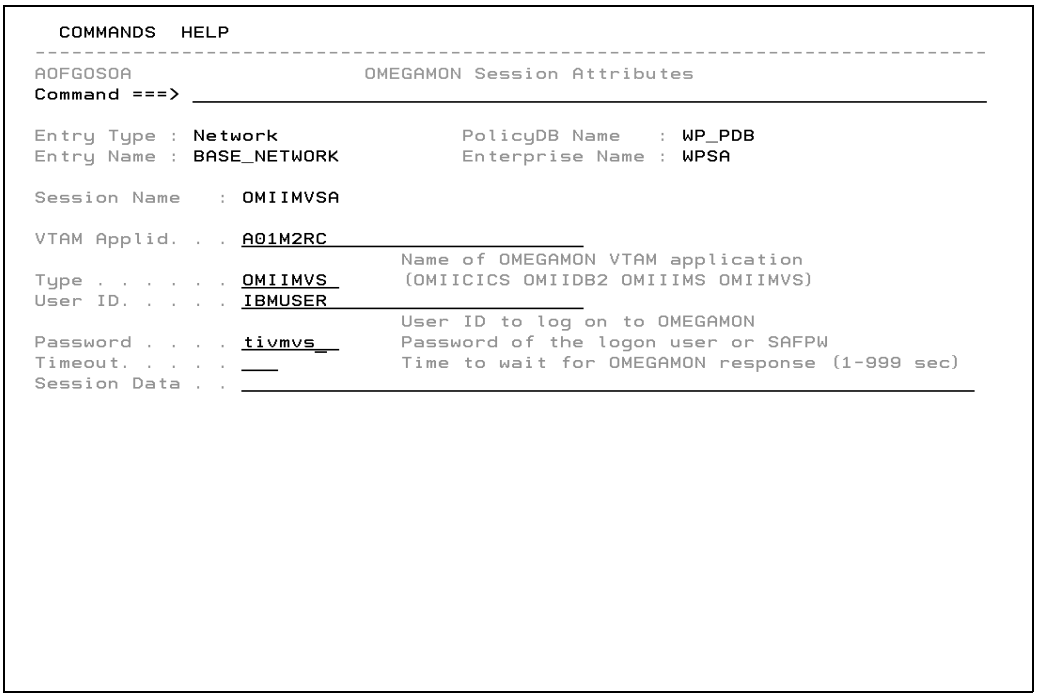
When you initially select this policy item there are no sessions defined.

This example shows two sessions defined, OMIIMVSA and OMIIMVSB:



In general, you will need a session between the Automation Agent and each OMEGAMON monitor (MVS, CICS, DB2, and IMS) you will retrieve exceptions from. The OMEGAMON monitor can be running on the same system as the agent or in a different system.

To define (or display) the session parameters **s**elect the session and press **Enter** to display the *OMEGAMON Session Attributes* panel (AOFGOSOA).



The *OMEGAMON Session Attributes* panel displays the session parameters used to connect to an OMEGAMON monitor:

- **VTAM Applid**: A01M2RC (VTAM APPLID of the OMEGAMON II data collector Application)
- **Type**: OMIIMVS (this is a session with OMEGAMON II for MVS)
- **User ID**: IBMUSER (OMEGAMON user ID)
- **Password**: tivmvs

  You can choose to use password encryption by entering **SAFPW** in the password field. If you choose password encryption you also need to define the AUTHENTICATION policy item. Password encryption requires the NetView password data set and is discussed in the section on security.

This example defines a session between the SA z/OS agent and an OMEGAMON for MVS monitor where A01M2RC is the VTAM APPLID of the OMEGAMON monitor. IBMUSER will be logged on with the password as shown.

The OMEGAMON monitor may be running on the same system as SA z/OS or it may be on a different system. If it is on a different system you will need to ensure that you have VTAM connectivity between the systems.

When you define monitor resources for OMEGAMON exceptions, SA z/OS will attempt to start the session when it initializes the monitor resource. If the OMEGAMON monitor Application is already active then the session will start successfully. However, there may be instances when the OMEGAMON monitor Application is not active when the session

start is attempted. This can cause an error that can only be corrected manually by the operator.

To avoid this scenario you can define a HasParent relationship between the SA z/OS monitor resource (MTR) and the OMEGAMON monitor Application. The HasParent relationship will ensure the session starts when the OMEGAMON monitor Application is started or active. The relationship will also ensure that the session is stopped when the OMEGAMON monitor Application is stopped. See *3.4 Defining Relationships* for further details.

3.1.2 Network Policy Definitions Summary

At this point you have defined the OMEGAMON SESSIONS policy items to connect an SA z/OS Automation Agent with two OMEGAMON for MVS monitors. The next step is to define the monitor resource policy.

3.2 Monitor Resource Policy (MTR) Definitions

Monitor resources are used to set the Health Status of Applications or Application Groups.

This section of the document will show you how to set the Health Status of a resource when an OMEGAMON for MVS exception is detected. These examples will use the XREP exception. Additionally, you will see how to define a command to be issued every time the exception is detected.

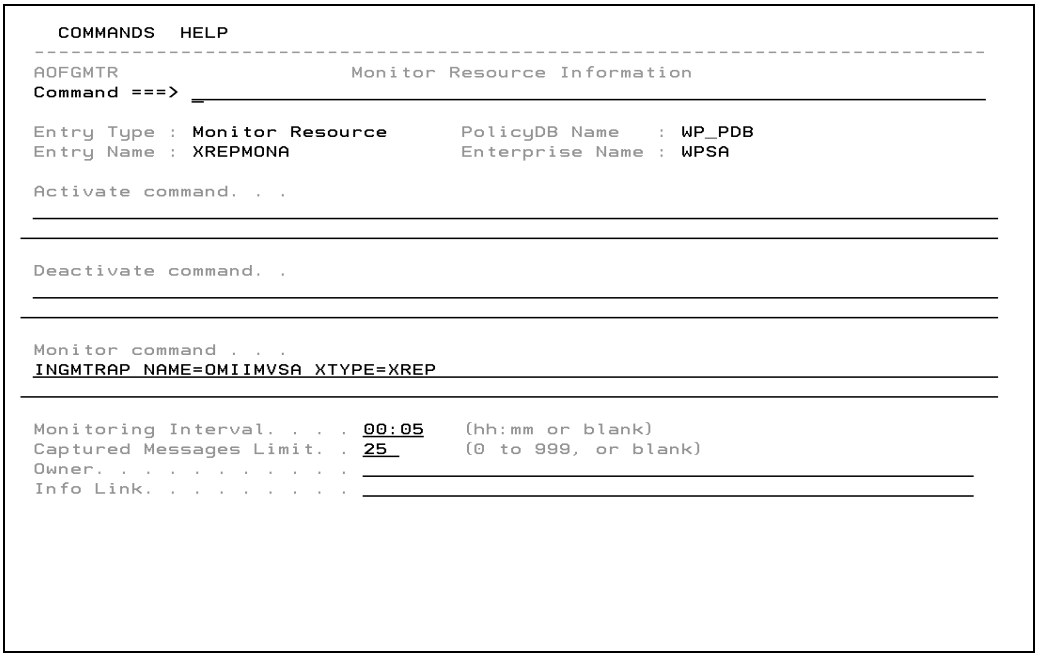
First, create a monitor resource. In this example it is called XREPMONA. The monitor resource will be interested in XREP exceptions on the MVSA (TIVED1) system.

Two tasks are required:

- Use the MONITOR INFO policy item to define the monitor routine and exception you are interested in.
- Use the MESSAGES/USER DATA policy item to define the Health Status setting when the exception is detected.

3.2.1 MONITOR INFO Policy Item

Select the MONITOR INFO policy item to display the *Monitor Resource Information* panel (AOFGMTR) to define, at minimum, the monitor routine, including the OMEGAMON exception, and a monitoring interval.



This example will cause the monitor routine (INGMTRAP NAME=OMIIMVSA XTYPE=XREP) to be scheduled every five minutes to retrieve XREP exceptions from the OMEGAMON for MVS monitor using the OMIIMVSA session that was defined earlier.

Each time the INGMTRAP routine is invoked it will request XREP exception data from the OMEGAMON for MVS monitor. If an exception is detected then an ING080I message will be generated.

Press **PF3** to save the monitor definition.

Note: The examples used in this document assume a one-to-one correlation between each session and its OMEGAMON monitor. This is not required. Each OMEGAMON monitor can have multiple sessions from the same agent.

3.2.2 MESSAGES/USER DATA Policy Item

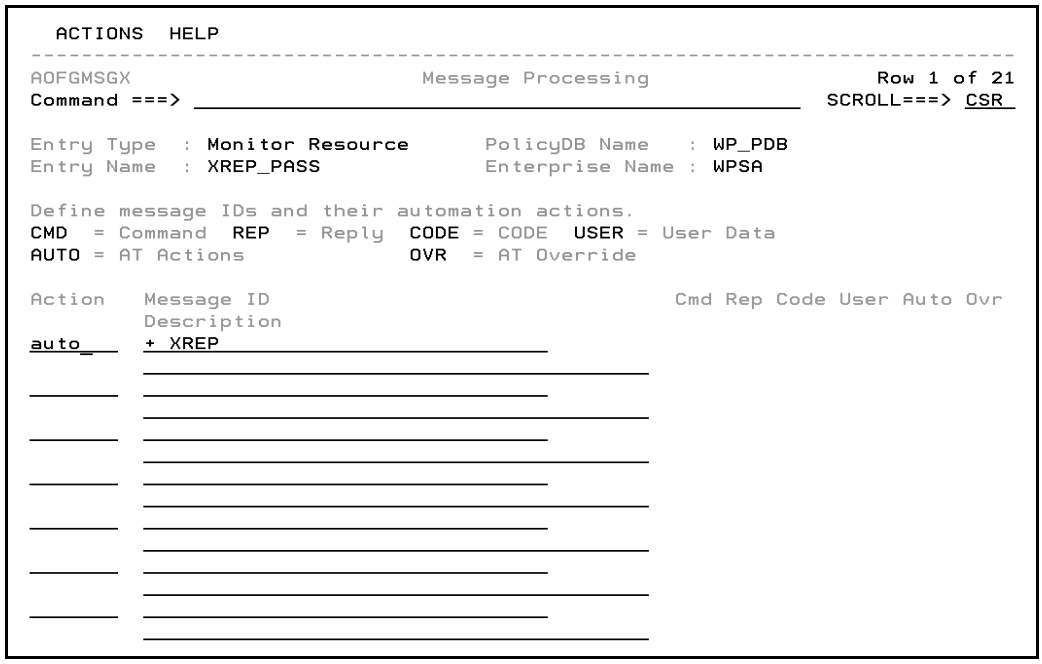
The MESSAGES/USER DATA policy item can be used to map the existence of an exception to a change in the SA z/OS Health Status. You can also define one or more commands to be issued when the exception is detected.

In this section you will see two examples. The first example will change the SA z/OS Health Status based on the existence of an OMEGAMON exception. The second example will change the SA z/OS Health Status based on the existence of an OMEGAMON exception and issue a command (MSG ALL).

More complex automation is possible and will be discussed later in this document when PASSes and CODEs are discussed.

3.2.2.1 Map the Existence of an Exception to a Health Status Change

The most basic function of monitoring for OMEGAMON exceptions is to map the existence on an exception to a change in the SA z/OS Health Status.



This is the *Message Processing* panel (AOFGMSGX) for a monitor resource named XREP_PASS. It is displayed when you select the MESSAGES/USER DATA policy. To map an exception to an SA z/OS Health Status:

- Use the *Message ID* field to define a six character message ID that includes the four character exception such as XREP preceded by a plus sign (+) and a blank.

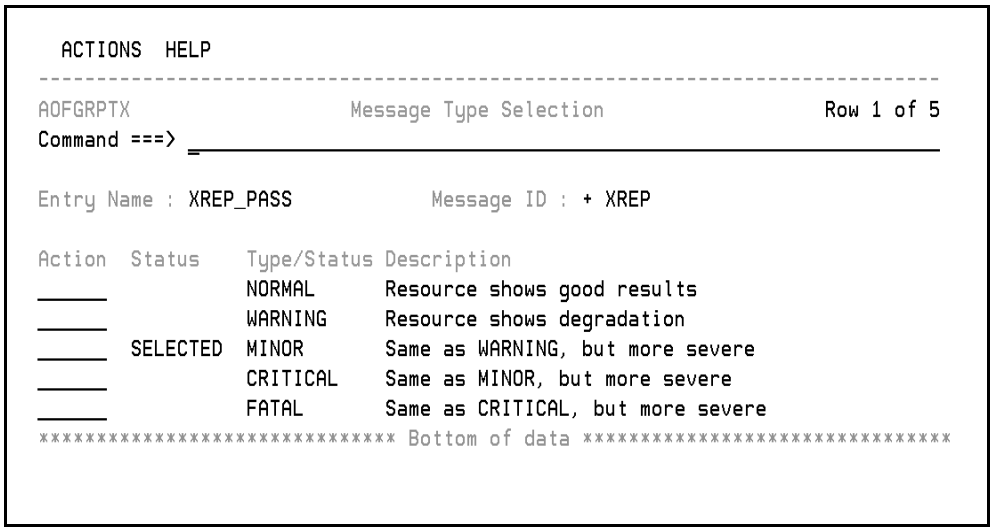
  + XREP, for example.

  All exceptions must be identified with a + as the first character and must followed by a blank.

  The + is used by SA z/OS to identify the message as an exception ID instead of an actual message ID.

- Enter **AUTO** for the *Action*.

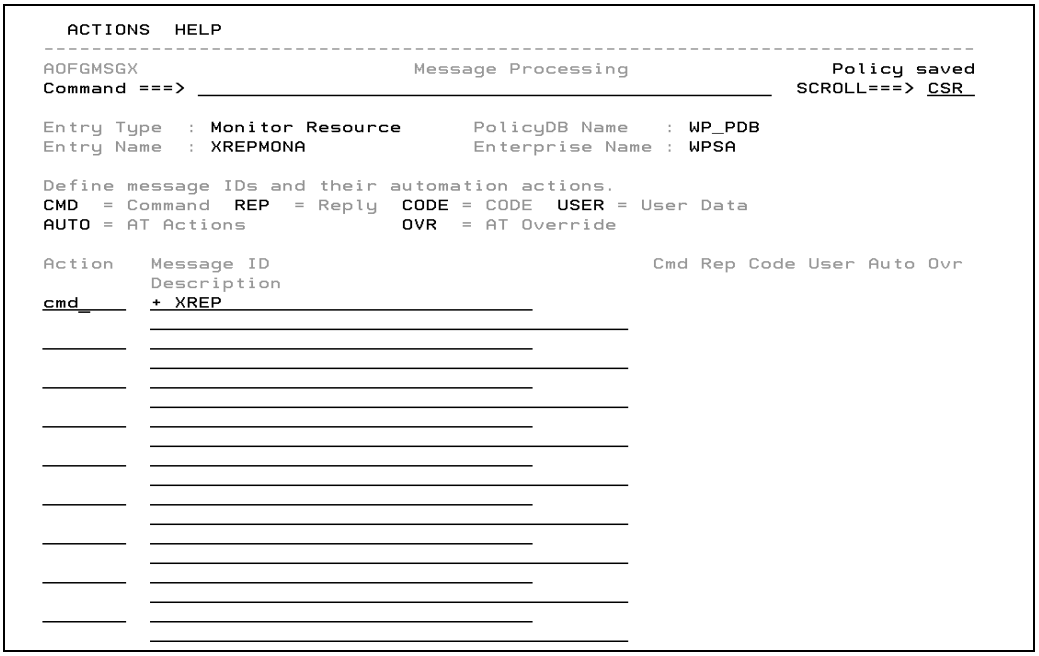
  An action of AUTO will display the *Message Type Selection* panel (AOFGRPTX) where you can define the Health Status. In this example, every time an XREP exception is detected the Health Status will be set to MINOR.



Press **PF3** to complete your selection. When the XREP_PASS monitor resource detects an XREP exception the Health Status will be set to MINOR, generating an ING080I message as part of the process.

3.2.2.2 Issuing Commands for Exceptions

This example will show how you can set the SA z/OS Health Status and issue one or more commands to provide automation for the OMEGAMON exception.



This is the *Message Processing* panel (AOFGMSGX) for a monitor resource named XREPMONA. It is displayed when you select the MESSAGES/USER DATA policy.To map the exception to a Health Status and issue a command:

- Use the *Message ID* field to define the exception, + XREP, for example.
- Enter **CMD** for the *Action*.

  An action of CMD will display the *CMD Processing* panel (AOFGMSGC) where you can define the Health Status setting and one or more commands to be issued every time the exception is detected. These commands can take the necessary actions to recover from the exception and perform additional notification, for example.

You can enter other actions such as CODE for more automation options. CODEs and PASSes will be discussed in the *Complex Automation* section of this document.

Note: Now that you have seen that the OMEGAMON exceptions are retrieved from the OMEGAMON monitor through the use of the INGMTRAP command you should understand that *exception is detected* means *exception is retrieved*.

```
  COMMANDS  HELP
-------------------------------------------------------------------------------
AOFGMSGC                        CMD Processing                    Row 1 of 20
Command ===> ____________________________________________________  SCROLL===> CSR

Entry Name : XREPMONA              Message ID : + XREP

Enter commands to be executed when resource issues the selected message.
or define this message as status message.

Status . . . WARNING      ('?' for selection list)

Pass/Selection Automated Function/'*'
Command Text
_________  ________
MSG ALL,XREP Exception detected on TIVED1_____________________________________
_____________________________________________________________________________

_________  ________
_____________________________________________________________________________
_____________________________________________________________________________

_________  ________
_____________________________________________________________________________
_____________________________________________________________________________

_________  ________
_____________________________________________________________________________
_____________________________________________________________________________
```

In this example, every time an XREP exception is detected, the Health Status will be set to WARNING and a MSG ALL command will be issued. You can enter a question mark (**?**) in the *Status* field to see a list of all possible Health Status values.

The command that is issued can be any valid NetView command. MSG ALL is only used as a generic example. In most implementations you will need to invoke a REXX routine to collect some more data and take corrective actions to provide a form of simple automation for XREP exceptions.

You can define multiple commands on this panel. The first may be to notify appropriate personnel of the exception. The second may be a routine to take corrective actions. More complex automation is available when you define PASSes or CODEs. These will be discussed in the *Complex Automation* section later in this document.

Press **PF3** to save your definitions.

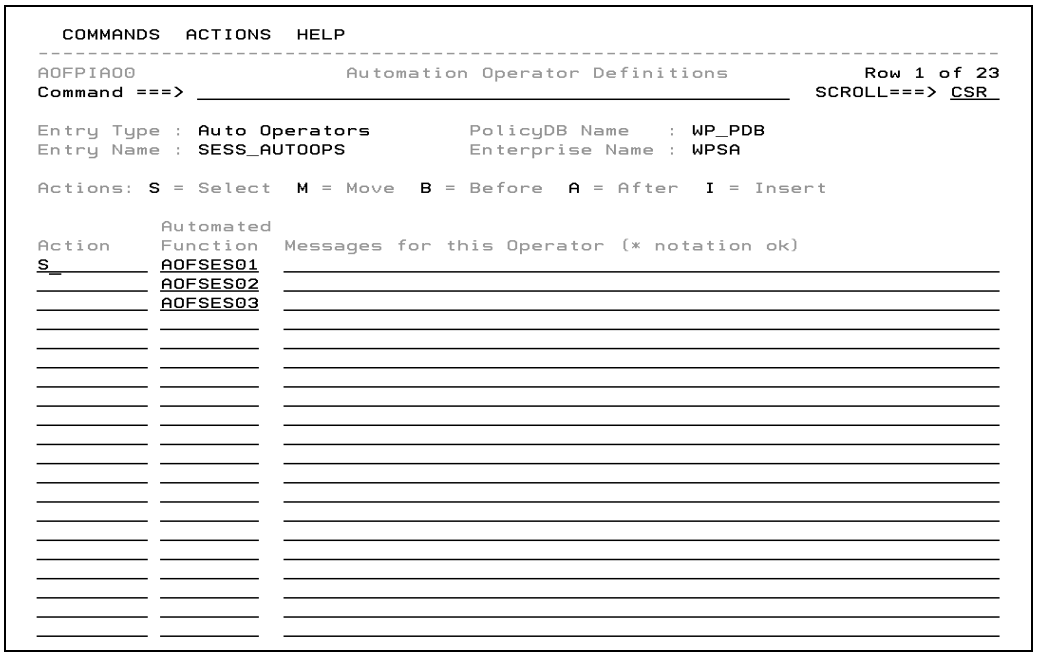
3.3 Automation Operator (AOP) Policy Definitions

When the *OMEGAMON add-on policies are imported make sure the SESS_AUTOOPS policy item is in the list of policies to import. SA z/OS uses SESS_AUTOOPS to provide three automation tasks (autotasks) to use for the sessions between the Automation Agent and the OMEGAMON monitors.

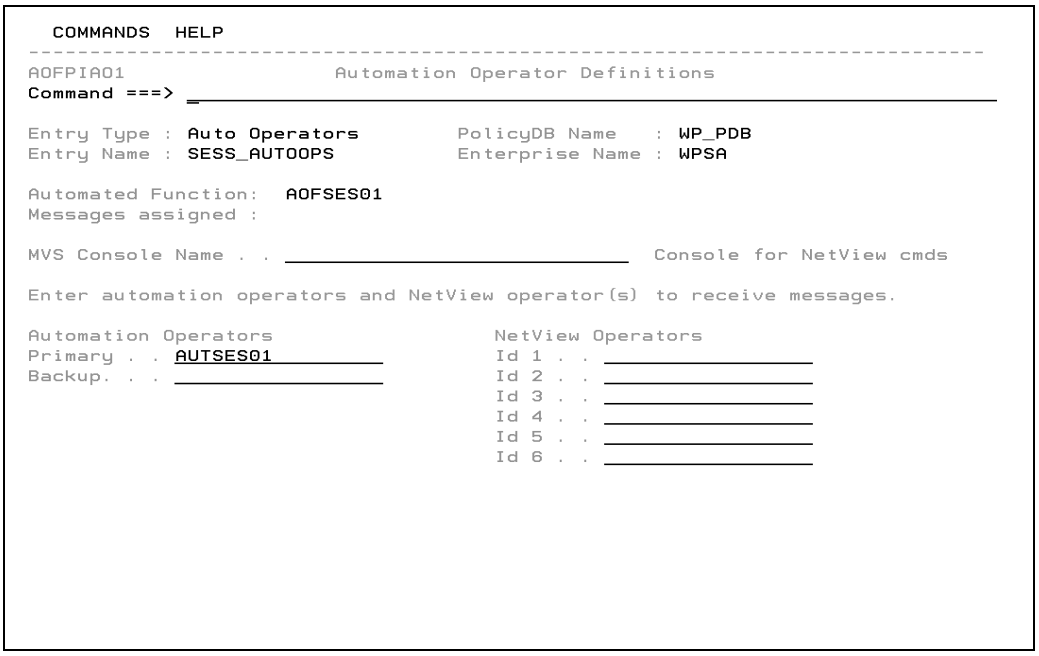
Select the AOP (Auto Operators, option **37**) policy to view or modify the SESS_AUTOOPS policy item. The *Policy Selection* panel (AOFGEPOL) will be displayed for the AOP policy.

```
  ACTIONS  HELP
------------------------------------------------------------------------------
AOFGEPOL                        Policy Selection                    Row 1 of 5
Command ===> _____________________________________________  SCROLL===> CSR

Entry Type : Auto Operators        PolicyDB Name   : WP_PDB
Entry Name : SESS_AUTOOPS          Enterprise Name : WPSA

Action      Policy Name          Policy Description
__________  DESCRIPTION         Enter description
s_________  OPERATORS           Define automation operators
__________  -------------------- ----------------------------------------------
__________  WHERE USED          List systems linked to this entry
__________  COPY                Copy data from an existing entry
****************************** Bottom of data ********************************
```

Select the OPERATORS policy item to see the automation functions.



The *Automation Operator Definitions* panel (AOFPIAO0) displays three automated functions for SESS_AUTOOPS. **S**elect an automated function (AOFSES01, for example) to see the actual autotask name.



AUTSES01 is one of the automation tasks defined to connect the Automation Agent to an OMEGAMON monitor. There are three automation tasks defined (AUTSES01 through AUTSES03) for the three automated functions.

If your installation requires different task names then you can modify the task names on the *Automation Operator Definitions* panel (AOFPIAO1). You would have to do this for each automated function. If you change the task name then you need to ensure you have the new task name defined in the operator definitions (DSIOPFU, for example).

The backup task name will be ignored if you specify one.

Suppose you were running all four OMEGAMON monitors (MVS, CICS, IMS, and DB2) and you require a session with each monitor. You would have four sessions defined. If you kept the default SESS_AUTOOPS policy definitions of three automation tasks then one task would be forced to connect to two of the OMEGAMON monitors.

Since automation tasks do not take much storage you may want to consider defining one automation task per session with an OMEGAMON monitor. In this case, define an additional automated function, AOFSES04, with an automation task of AUTSES04. SA z/OS provides operator definitions for AUTSES01 through AUTSES10 so you would not need to modify the operator definitions.

3.4 Defining Relationships

This section is very important.

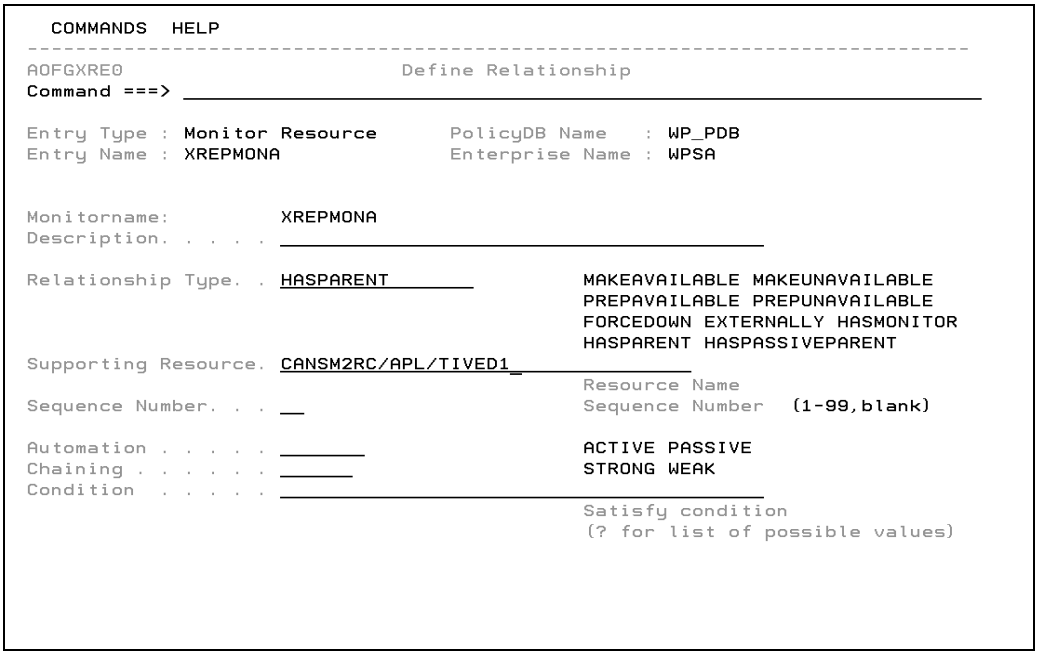
As was discussed earlier, you should define a HasParent relationship between the monitor resource and its OMEGAMON monitor Application. If not, you could run into a scenario where SA z/OS attempts to start the session before the Application is active. The session start will fail and the status of the session will be SESSFAIL. An operator must take action to change the status of the session before it can be restarted. Sessions can only be started if their status is MAINT or INACTIVE. The monitor resources will also fail and will need to be reset when the session is activated.

To avoid this scenario you should define a HasParent relationship between the monitor resource and the OMEGAMON monitor Application. You can define a HasParent relationship or both a MakeAvailable and a MakeUnavailable relationship.

3.4.1 Monitor Resource RELATIONSHIPS Policy Item

Select the RELATIONSHIPS policy item for the monitor resource (for example, XREPMONA) and define a HasParent relationship.

This is an example of a HasParent relationship between XREPMONA and the OMEGAMON for MVS monitor (CANSM2RC):



CANSM2RC is the name of the OMEGAMON for MVS monitor Application. It was imported from the *OMEGAMON add-on as OMIIM2RC and renamed during the flat file update.

In this case, the session will be with the OMEGAMON for MVS monitor running on TIVED1 **only** so a fully qualified name is shown for the *Supporting Resource* field.

Press **PF3** to save the relationship.

The HasParent relationship between the SA z/OS monitor resource and the OMEGAMON monitor Application will synchronize the start and stop of both resources, saving you a lot of time and preventing problems.

3.5 TAF SRCLU Definitions

The session between the Automation Agent and each OMEGAMON monitor will use the NetView terminal access facility (TAF) in full-screen mode. TAF requires VTAM APPL definitions for its SRCLUs. Define *dynamic* SRCLU APPL definitions for each session similar to:

```
TFDA#000 APPL  MODETAB=AMODETAB,EAS=9, X
               DLOGMOD=M2SDLCNQ
```

Where **DA** is the last two characters of your NetView domain name, AOFDA in this example. The DLOGMOD parameter identifies this as a standard 3270 session with a 24x80 screen size.

Your system administrator should have done this customization during the installation of NetView.

If you encounter problems connecting to the OMEGAMON monitor you can try to issue a NetView **BGNSESS** command similar to this one to start the session:

```
BGNSESS FLSCN,APPLID=A01M2RC
```

If the session fails when SA z/OS attempts to start it but starts when you enter a BGNSESS command then you have an error in your SA z/OS policy definitions.

4 SA z/OS Operator Commands

Up to this point you have performed the administrative tasks to implement performance driven automation with OMEGAMON and SA z/OS.

In this section the focus turns to the operational commands provided by SA z/OS to:

- Manage the sessions between the Automation Agent and the OMEGAMON monitors using the INGSESS command.
- Issue OMEGAMON commands directly from the Automation Agent using the INGOMX command.
- Retrieve OMEGAMON exceptions using the INGOMX command.
- Write your own monitor routine using the INGOMX command.
- Display monitor resources with the DISPMTR command.
- Reset the Health Status and monitoring using the INGMON command.
- Issue OMEGAMON minor commands using the INGOMX command in a REXX EXEC.
- Pass parameters to the OMEGAMON commands being invoked when using the INGOMX command.

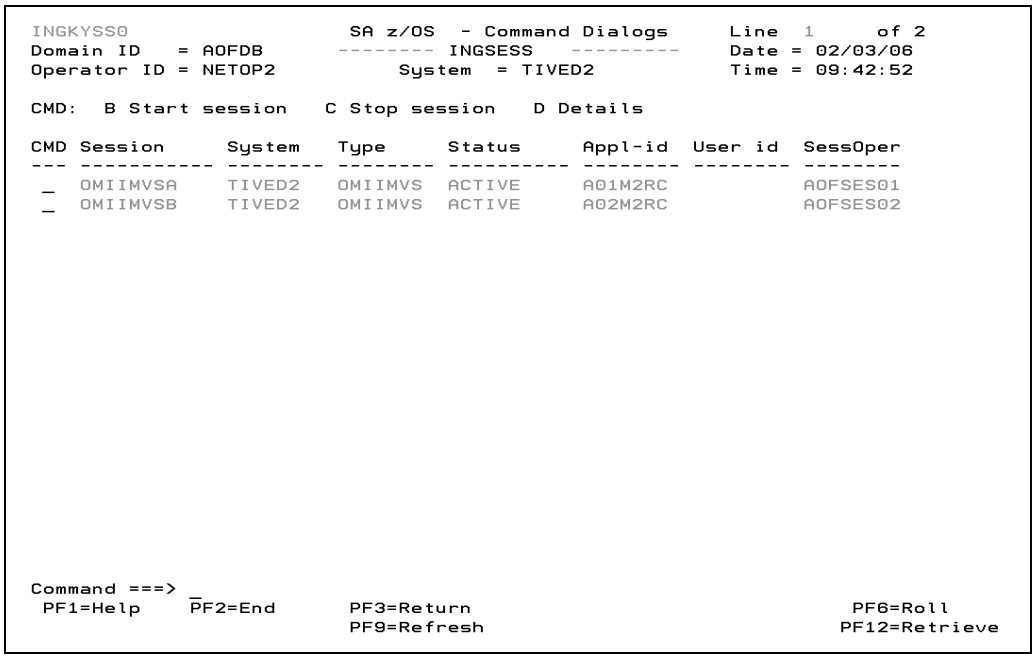
4.1 Manage Sessions (INGSESS)

SA z/OS provides the INGSESS command to manage the sessions between the agent and the OMEGAMON monitors. INGSESS can be used to:

- Display sessions.
- Display detailed session data, including statistics for the session.
- Start or stop sessions.

4.1.1 Display Sessions

The INGSESS command can be used to display one or more sessions. INGSESS without any parameters will display all sessions. For example:

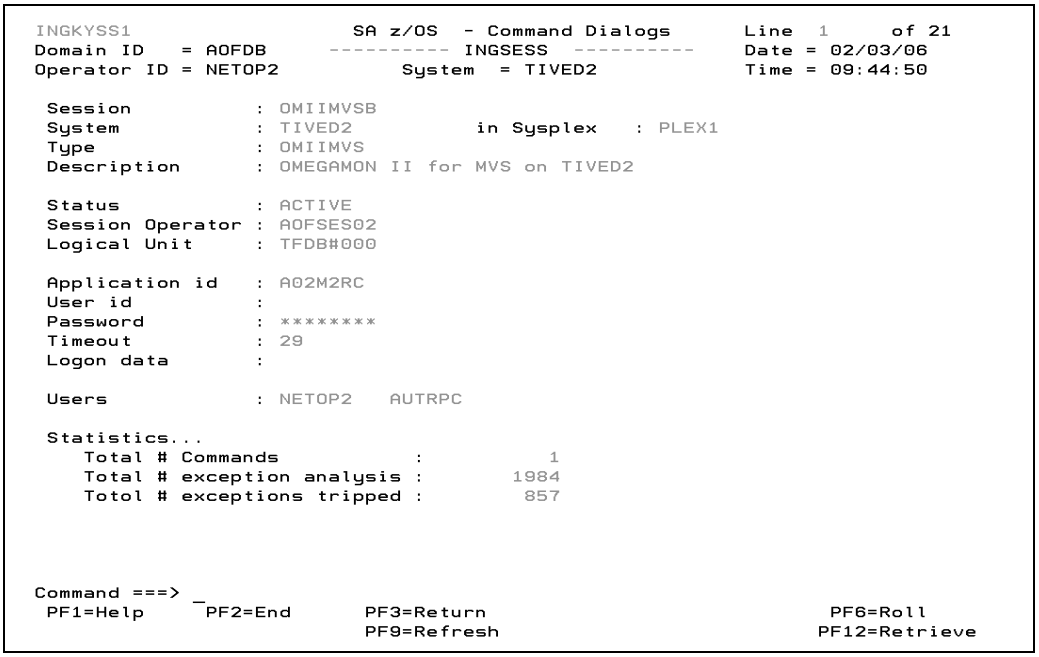


This example shows two sessions defined. OMIIMVSA is a session with A01M2RC using the AOFSES01 (AUTSES01) automation operator. OMIIMVSB is the session with A02M2RC using AOFSES02 (AUTSES02) as the automation operator. Both sessions are active.

You can display a subset of the sessions. For example, **INGSESS OMIIMVSA** would display only the OMIIMVSA session.

4.1.2 Display Detailed Session Data

Option **D** from the INGSESS panel will display session details, including statistics. For example, tab to OMIIMVSB and enter **D** and you will see a panel similar to:



The INGSESS details panel displays more detailed information for a session:

- **Type=OMIIMVS:** This is a session to an OMEGAMON II for MVS monitor.
- **Session Operator=AOFSES02:** The automation operator is AUTSES02.
- **Logical Unit=TFDB#000:** The TAF full-screen SRCLU.
- **Application id=A02M2RC:** The VTAM APPLID of the OMEGAMON II for MVS monitor.
- **User id:** This is the ID used to log on to the OMEGAMON monitor. In this example the user ID is null. OMEGAMON will support a connection without a user ID. However, you will most likely have limited function due to security.
- **Password=********:** The password for the session.
  - If **SAFPW** then encrypted passwords are being used.
  - If ******** then encrypted passwords are not being used. The ******** is displayed instead of the actual password.
- **Users:** List of tasks that have used this session. For example:
  - NETOP2: Operator who issued one or more INGOMX commands.
  - AUTRPC: Autotask that is used by SA z/OS monitor resource processing to issue the INGMTRAP command.
- **Total # Commands:** Number of times an INGOMX CMD= command has been issued. In this example INGOMX has been called one time to issue an OMEGAMON command such as CSAA.

- **Total # exception analysis:** Number of times exception retrieval has been requested. This includes calls from INGMTRAP as well as INGOMX TRAP XTYPE=. In this example, there has been 1,984 requests to retrieve an exception.
- **Total # exceptions tripped:** Number of exceptions returned from exception analysis. In this example, 857 exceptions have been detected.

In summary, for this example, one OMEGAMON command was issued from NETOP2. There were 1,984 requests to retrieve an exception from the OMEGAMON monitor. These include requests from the monitor resources (INGMTRAP) as well as operators (INGOMX XTYPE=). From the 1,984 requests there have been 857 exceptions detected (retrieved).

INGSESS OMIIMVSB REQ=DETAIL will also display the same panel.

4.1.2.1 Notes about Exception Detection

Here are some helpful notes related to exceptions and detailed session statistics:

- Not every exception request will result in an exception detected.
- Not all exceptions detected will generate an ING080I message.
  - For monitor resources using INGMTRAP an ING080I message will be generated for each exception.
  - For direct (operator command or REXX routine) calls to INGOMX XTYPE= the exception is returned but no ING080I message is issued.
- One exception request may generate a no exceptions detected message (ING081I), one exception detected message (ING080I), or multiple exception detected messages (ING080I). In this example there were 857 exceptions detected. There is no way of knowing (from this panel) if any one request generated multiple exceptions.
- These statistics are reset when the session is started.

4.1.3 Start or Stop Sessions

INGSESS can also be used to start or stop a session.

Sessions can only be started if the session status is INACTIVE or MAINT. Attempting to start a session in any other state will result in an error.

For example, suppose the session with OMIIMVSB has failed. Its status would be SESSFAIL.

```
INGKYSS0                    SA z/OS  - Command Dialogs      Line  1     of 2
Domain ID   = AOFDB        -------- INGSESS   ---------     Date = 02/03/06
Operator ID = NETOP2            System  = TIVED2            Time = 11:53:05

CMD:  B Start session   C Stop session   D Details

CMD Session     System   Type     Status     Appl-id  User id  SessOper
--- ----------- -------- -------- ---------- -------- -------- --------
 _  OMIIMVSA    TIVED2   OMIIMVS  ACTIVE     A01M2RC           AOFSES01
 _  OMIIMVSB    TIVED2   OMIIMVS  SESSFAIL   A02M2RC           AOFSES02









Command ===> _
 PF1=Help     PF2=End      PF3=Return                          PF6=Roll
                           PF9=Refresh                       PF12=Retrieve
```

If you attempt to start the session you will see:

```
ING001I INGOMX SERVICE FAILED, RC=9, REASON=ATTACHSESSION
OMIIMVSB
```

RC=9 tells you that the session is not in a startable state.

You need to correct the error before trying to restart the session. Perhaps the OMEGAMON monitor Application is not active.

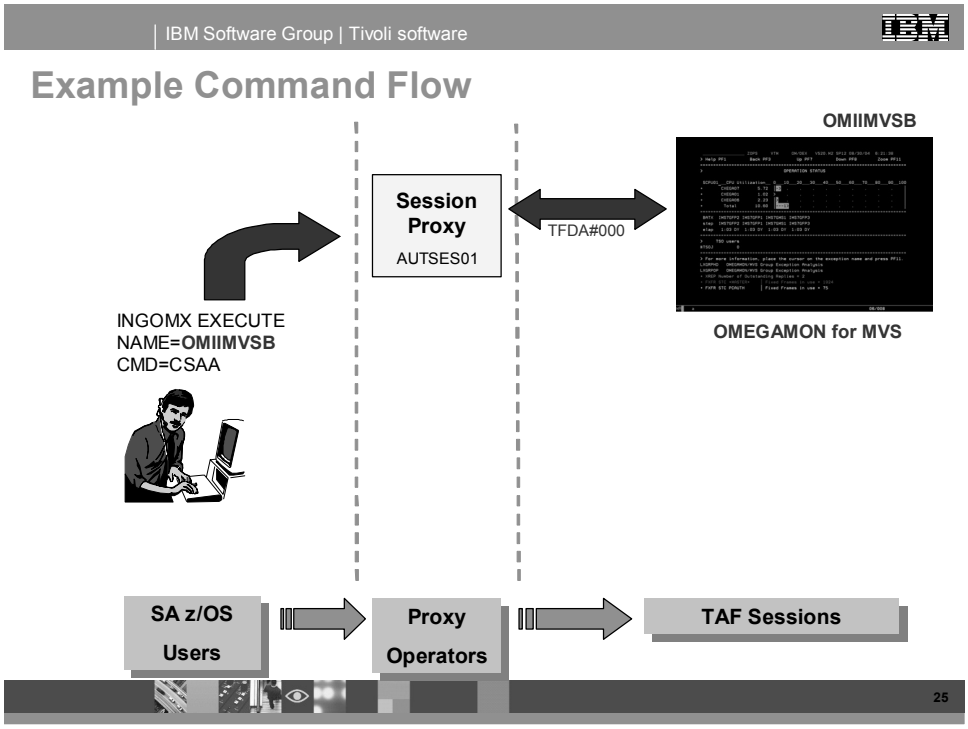
After you correct the error, you must choose option **C** to stop the session. This will change its status to MAINT which is one of the startable states. When the session status is MAINT you can choose option **B** to start the session.

You can also use **INGSESS OMIIMVSB REQ={START | STOP}**.

Starting a session resets the command and exception statistics that are being tracked.

4.2 Issue OMEGAMON Commands (INGOMX)

You can issue OMEGAMON (major, minor, and immediate) commands over the sessions. This slide illustrates the flow involved.



The INGOMX command is used to issue OMEGAMON commands from the agent. The operator routes the command to the session identified by the NAME= parameter. SA z/OS will use the NAME= parameter to route the request to the *proxy operator* (AUTSES01) assigned to the session (OMIIMVSB).

The operator is unaware of which proxy operator is used, which OMEGAMON user is logged on, or if the session is even active. The operator only needs to enter the INGOMX command and know where they want the command sent. If the session is not active the agent will attempt to start it.

The response will be returned back to the operator as if they were actually connected to the OMEGAMON monitor.

Note: This flow also applies to the use of the INGMTRAP or INGOMX TRAP XTYPE= commands.

Here is an example of an OMEGAMON CSAA response:

```
CNMKWIND OUTPUT FROM  INGOMX EXECUTE CMD=CSAA NAME=OMIIMVSA  LINE 0 OF 9
*------------------------------ Top of Data -------------------------------*
 CSAA   SUMMARY
+
+          System
+         Maximum  Pre-CSAA   Orphan       Usage
+         -------  --------  -------  ---------------0__2__4__6__8__100
+    CSA     4768K        0         0     247K    5.2%|>                 |
+   ECSA   123548K        0       79K   26646K   21.6%|--->               |
+    SQA     2476K        0       128     544K   22.0%|--->               |
+   ESQA    12544K        0        3K    9723K   77.5%|-------------->    |
*------------------------------ Bottom of Data ----------------------------*
```

If you are familiar with NetView then you may have noticed that the output in this example was captured using the WINDOW command. Using WINDOW greatly enhances the readability of output from commands such as CSAA where column formatting must be precise.

You can also issue the INGOMX command within a NetView PIPE to improve the readability of the response:

```
NetView V5R2 SA V3       Tivoli NetView   AOFDA NETOP1   02/01/06 16:47:34 A
* AOFDA    PIPE NETV INGOMX EXECUTE CMD=CSAA NAME=OMIIMVSA | COLL | CONS ONLY
| AOFDA
 CSAA   SUMMARY
+
+          System
+         Maximum  Pre-CSAA   Orphan       Usage
+         -------  --------  -------  ---------------0__2__4__6__8__100
+    CSA     4768K        0         0     247K    5.2%|>                 |
+   ECSA   123548K        0       79K   26646K   21.6%|--->               |
+    SQA     2476K        0       128     544K   22.0%|--->               |
+   ESQA    12544K        0        3K    9721K   77.5%|-------------->    |
--------------------------------------------------------------------------------
```


4.3 Retrieve OMEGAMON Exceptions (INGOMX)

The INGOMX command can also be used to retrieve exception data from an OMEGAMON monitor. The flow is exactly the same as when INGOMX is used to issue OMEGAMON commands. The operator will enter the INGOMX command. It will be routed to a proxy operator assigned to the TAF sessions with the specified OMEGAMON monitor.

INGOMX will retrieve the exception and return zero, one, or more OMEGAMON exception messages. If there are no exceptions you will not see a message. This is not an error.

For example, suppose you want to retrieve OMEGAMON SWPC (excessive swap count) exceptions from the OMEGAMON for MVS monitor running on the system known as MVSA (OMIIMVSA):

```
Command: INGOMX TRAP XTYPE=SWPC NAME=OMIIMVSA

Response:
+ SWPC STC BPXOINIT    | Excessive Swap counts = 80
+ SWPC STC OSNMPD      | Excessive Swap counts = 8
+ SWPC STC SNMPQE      | Excessive Swap counts = 13
```

This tells us that three address spaces had an excessive swap count.

4.4 Write Your Own Monitor Routine with INGOMX

Suppose you want to monitor for multiple exceptions. For example, the OMEGAMON for MVS WAIT **and** SWPC exceptions. A WAIT exception occurs when an address space has been waiting longer than *n* seconds. Furthermore, suppose that you are only concerned with the overlap between the two sets of exception messages.

The previous section provided you with an example of what an SWPC exception response looks like. Here is an example WAIT exception response. It may also contain many lines of data.

```
Command: INGOMX TRAP XTYPE=WAIT NAME=OMIIMVSA

Response:
+ WAIT STC RASP        | Wait: 20:42 MN
+ WAIT STC ANTMAIN     | Wait:   14 SEC
+ WAIT STC ANTAS000    | Wait: 20:03 MN
+ WAIT STC IEFSCHAS    | Wait: 20:50 MN
+ WAIT STC INETD4      | Swap: 16:26 MN  Det-Wait
+ WAIT STC FTPD1       | Swap: 18:55 MN  Det-Wait
+ WAIT STC NAMED3      | Wait:  4:32 MN
+ WAIT STC AUTOMGR     | Wait:   16 SEC
```

```
+ WAIT STC AUTOSSI     | Wait: 19:26 MN
+ WAIT STC TNF         | Wait: 20:44 MN
+ WAIT STC RESOLVER    | Wait: 20:03 MN
+ WAIT STC SYSLOGD8    | Wait:   10 SEC
+ WAIT STC CANSCN      | Wait:  3:19 MN
+ WAIT STC RXSERVE     | Swap: 18:55 MN  Det-Wait
+ WAIT STC OSNMPD      | Swap: 15:57 MN  Det-Wait
+ WAIT STC SNMPQE      | Swap: 16:03 MN  Det-Wait
+ WAIT STC APPC        | Wait:  3:18 MN
+ WAIT STC DLF         | Wait: 17:37 MN
+ WAIT STC RV04        | Swap: 17:35 MN  Det-Wait
+ WAIT STC RV03        | Swap: 17:35 MN  Det-Wait
+ WAIT STC CANSETE     | Wait:  7:59 MN
+ WAIT STC CANSM2      | Wait:   12 SEC
+ WAIT STC CANSM2HD    | Swap: 17:29 MN  Det-Wait
```

If you were to define a monitor resource to monitor for WAIT exceptions you would see an ING080I message for each line in the response. However, it may not a problem if some of the Applications are waiting. You would not want the ING080I messages to be issued for those Applications.

As stated earlier, suppose you are only concerned with the Applications with both an excessive wait time (WAIT) **and** an excessive swap count (SWPC). You would need to write your own monitor routine to retrieve both exceptions (SWPC and WAIT) from the OMEGAMON monitor and correlate the Applications that appear in both exception responses.

The SWPC and WAIT examples shown here have at least one Application in common, OSNMPD. The SWPC and WAIT exception messages for OSNMPD are:

```
+ SWPC STC OSNMPD      | Excessive Swap counts = 8
+ WAIT STC OSNMPD      | Swap: 15:57 MN  Det-Wait
```

The third token in each exception message is the *STC name*. For the purposes of discussion, assume the *STC name* is also the SA z/OS resource name. One possible design for your monitor routine (instead of using INGMTRAP) could be:

- Use INGOMX to retrieve the SWPC exceptions.
- Use INGOMX to retrieve the WAIT exceptions.
- Determine the Applications that are waiting and have excessive swap counts.
- Issue an INGMON command to set the Health Status for the monitor resource.
- Issue one or more INGSET commands to set the Health Status for the Applications that have both SWPC and WAIT exceptions.

  Refer to the WAITSWPC REXX EXEC in Appendix A for the exact details of the INGSET command and how it is built.

When the monitor routine is run it will need to set the Health Status for the monitor resource directly with an INGMON command and the Health Status of the Application with the INGSET command. Since the monitor routine is setting the Health Status it should end with a return code of eight (8) which tells the SA z/OS processes not to set the Health Status.

Note: The return code of eight is new for SA z/OS 3.1 in support of automation based on OMEGAMON exceptions.

If you are familiar with NetView PIPEs coding techniques you can retrieve both sets of exception messages with one INGOMX command:

```
PIPE NETV INGOMX TRAP XTYPE=(WAIT,SWPC) NAME=OMIIMVSA | CORR | and
so on.
```

The response would contain messages for each exception on an Application basis. The Applications that have exceptions for both SWPC and WAIT will look similar these messages where the exceptions are grouped together:

```
+ SWPC STC INETD4      | Excessive Swap counts = 55
+ WAIT STC INETD4      | Swap:  2:03 DY  Long-Wat

+ SWPC STC FTPD1       | Excessive Swap counts = 54
+ WAIT STC FTPD1       | Swap:  2:03 DY  Long-Wat

+ SWPC STC RXSERVE     | Excessive Swap counts = 54
+ WAIT STC RXSERVE     | Swap:  2:03 DY  Long-Wat

+ SWPC STC OSNMPD      | Excessive Swap counts = 61
+ WAIT STC OSNMPD      | Swap:  2:03 DY  Long-Wat

+ SWPC STC SNMPQE      | Excessive Swap counts = 65
+ WAIT STC SNMPQE      | Swap:  2:03 DY  Long-Wat
```

Assume the monitor resource name is WAIT_SWPC for the remaining discussion.

4.5 Display Monitor Resources (DISPMTR)

The DISPMTR command can be used to display and manage monitor resources. DISPMTR is an existing SA z/OS command. Since you can define monitor resources to monitor for OMEGAMON exceptions you may have a need to display information about the monitor resource.

For example, **DISPMTR WAIT_SWPC** will display the monitor information for the monitor resource that was defined to correlate SWPC and WAIT exceptions:

```
INGKYMO0                    SA z/OS  - Command Dialogs      Line  1     of 1
Domain ID   = AOFDA       -------- DISPMTR   ---------     Date = 03/14/06
Operator ID = NETOP1           Sysplex = SYSPLEX1           Time = 14:02:24

CMD: A Reset   B Start   C Stop   D Details   E INGVOTE  F INGINFO  I INGSCHED

CMD Monitor       System    Status         Health       Last monitored
--- ------------ --------  ------------  ------------ -------------------
 _  WAIT_SWPC    TIVED1    ACTIVE         WARNING      2006-03-14 14:01:58









Command ===> _
 PF1=Help     PF2=End       PF3=Return                                PF6=Roll
                            PF9=Refresh PF10=Previous PF11=Next     PF12=Retrieve
```

DISPMTR shows the monitor resource (WAIT_SWPC) is active with a Health Status of WARNING. You can press **PF11** to scroll to the right or select CMD **D** to view details about the monitor resource.

The DISPMTR details panel will look similar to:

```
INGKYMO1                    SA z/OS  - Command Dialogs      Line  1     of 75
Domain ID   = AOFDA       ---------- DISPMTR  ----------    Date = 03/14/06
Operator ID = NETOP1           Sysplex = SYSPLEX1           Time = 14:10:17

 Monitor          : WAIT_SWPC/MTR/TIVED1
 System           : TIVED1
 Description      : Combined MTR

 Commands...
    Activate      :
    Deactivate    :
    Monitoring    : WAITSWPC NAME=OMIIMVSA

 Interval         : 00:03

 Monitor Status   : ACTIVE  at  2006-03-14 14:08:01
 Health Status    : WARNING
                    CORRELATED EXCEPTIONS FOUND FOR: INETD4 OSNMPD SNMPQE

 History (maximum is 100) ...
                      -  INACTIVE       HEALTH=UNKNOWN
                    2006-03-14 12:49:30  -  ACTIVE         HEALTH=UNKNOWN
                    Monitor started
                    2006-03-14 12:54:25  -  ACTIVE         HEALTH=WARNING
                    STATUS SET BY WAITSWPC
                    2006-03-14 12:55:00  -  ACTIVE         HEALTH=WARNING
                    STATUS SET BY WAITSWPC
                    2006-03-14 12:58:01  -  ACTIVE         HEALTH=WARNING

Command ===> _
 PF1=Help     PF2=End       PF3=Return                          PF6=Roll
              PF8=Forward   PF9=Refresh                        PF12=Retrieve
```

The DISPMTR details panel tells you the monitor routine is **WAITSWPC NAME=OMIIMVSA**. It will be scheduled every three minutes. The current Health Status of the monitor resource is WARNING. The monitor routine found one or more

Applications in both the SWPC and WAIT exception responses. In fact, the monitor routine was coded to issue a message to inform the operators exactly which resources are affected:

CORRELATED EXCEPTIONS FOUND FOR: **INETD4 OSNMPD SNMPQE**

The Health Status and message in this example were set by:

INGMON *WAIT_SWPC/MTR/TIVED1*,STATUS=*WARNING*,MSG=*'CORRELATED EXCEPTIONS FOUND FOR: INETD4 OSNMPD SNMPQE'*

The WAITSWPC routine ended with a return code of eight to tell SA z/OS processing that the Health Status was being set by the monitor routine. The SA z/OS monitor resource processes the return code and takes no action except to reschedule the monitor routine based on the monitor interval definition.

The WAITSWPC routine is provided later in this document, as a reference.

4.6 Reset Health Status and Monitoring (INGMON)

Up to this point you have learned how to define monitor resources to retrieve OMEGAMON exceptions and set the Health Status based on the exception. You have also learned how to drive automation when the exception is detected. Your automation should take actions to correct the problem.

Once the problem is corrected by automation the Health Status should be reset to NORMAL to indicate the problem no longer exists.

This can be accomplished with an INGMON command:

INGMON *monitor_name* MSGTYPE=*four_char_exception_ID* CLEARING=YES STATUS=*health_status* MSG='*descriptive_text*'

- Monitor_name is the name of the SA z/OS monitor resource such as WAIT_SWPC.
- The MSGTYPE parameter must be an OMEGAMON exception such as XREP.
- CLEARING=YES will reset the PASS count and DISABLETIME processing.
- The STATUS parameter is required. Since this is a *clearing event* you will most likely want to set the Health Status to NORMAL or UNKNOWN. The Health Status will be set the next time the monitor routine is executed.
- The MSG parameter is optional. It can be used to identify that the problem was corrected and the status was reset.

For example:

INGMON XREPMONB MSGTYPE=XREP CLEARING=YES STATUS=NORMAL MSG='Clear XREPMON monitor status'

This INGMON command will set the Health Status of the XREPMONB monitor resource to NORMAL and set the message displayed on the DISPMTR panels to: *Clear XREPMON monitor status.*

CLEARING=YES will also reset any PASS counts or DISABLETIME processing.

In the case of the example WAIT_SWPC monitor resource you should also code INGSET commands for each of the Applications to reset their status. See the WAITSWPC routine for more information.

4.7 Issue OMEGAMON Minor Commands

So far you have seen how to issue OMEGAMON commands such as CSAA. Several OMEGAMON major commands support minor commands. The major commands must be issued first and then followed immediately by the minor commands. To accomplish this INGOMX can be called with a list of commands in the default (PIPE) safe. For example,

```
/* Issue SYS major command, ahead of minor commands */
cmd.1 = "CMD=SYS"

/* Issue first minor command: */
/* FCSA (CSA frames below 16M) */
cmd.2 = "CMD=FCSA"

/* Issue next minor command: */
/* FCOM (CSA, LPA, SQA, and nucleus below 16M) */
cmd.3 = "CMD=FCOM"

/* Issue subsequent minor commands: */
cmd.4 = "CMD=FECS" /* Minor: extended CSA frames */
cmd.5 = "CMD=FELS"  /* Minor: extended LSQA frames */
cmd.6 = "CMD=FPLP"  /* Minor: PLPA frames */
cmd.7 = "CMD=FTOT"  /* Minor: total frames */

/* Set index for CMD. array for the commands */
cmd.0 = 7


/* Issue PIPE command. Place CMD. contents in default safe
and call INGOMX with CMD=* */
'PIPE STEM cmd. COLLECT',
  ' | NETV INGOMX EXECUTE,NAME=OMIIMVSA,CMD=*',
  ' | COLLECT ',
  ' | CONS ONLY'
```


Executing this REXX EXEC would produce output similar to:

```
SYS    >> WLM Goal mode OPT=00 SYSRES=(G1601F,0400) <<
fcsa         70      280 K
fcom        479     1916 K
fecs       5634    22536 K
fels       5353    21412 K
fplp        204      816 K
ftot     131072   524288 K
```

SYS is the major command and each of the minor commands (fcsa, fcom, fecs, fels, fplp, and ftot) are issued after the major command. The minor commands can only be issued after the major command.

If you are not familiar with writing NetView PIPEs see the *IBM Tivoli NetView for z/OS Programming: PIPEs* manual.

4.8 Use OMEGAMON Command Parameters

Some OMEGAMON commands require additional parameters to be passed along with the command. You can pass the command parameters using the INGOMX PARM keyword.

For example, the OMEGAMON for MVS SVOL command can be used to display available space on a specific disk. SVOL requires the volume serial (VOLSER) of the disk:

INGOMX EXECUTE CMD=SVOL **PARM=TVED12** NAME=OMIIMVSA

This would produce output similar to:

```
>SVOL  TVED12 0501 STR/RSNDT FREE(00976,00059) AREAS=0012
MAX_CNTG(00835,00000)
```

Some OMEGAMON commands will provide additional data when parameters are specified. For example, to issue a CSAA USAGE command to display the users of CSA, ECSA, SQA, or ESQA:

INGOMX EXECUTE CMD=CSAA **PARM=USAGE** NAME=OMIIMVSA

The result would look similar to:

```
CNMKWIND OUTPUT FROM  INGOMX EXECUTE CMD=CSAA PARM=USAGE  NA  LINE 0 OF 22
*------------------------------ Top of Data -------------------------------*
_CSAA   USAGE   AREA(CSA)  BOUND(0)
+   Jobname   Asid      Usage                    CSA Usage By Job
+ ----------- ----  -------------0___4___8__12__16__20__24__28__32__36__40
+ *SYSTEM*    0000     82K   1.7% >
+ *MASTER*    0001     77K   1.6% >
+ JES2        001D     33K    .7%
+ RACF        0024     32K    .7%
+ VTAM        001E     11K    .2%
+ AUTOSSI     0022      4K    .1%
+ CANSM2HI    0040      3K    .1%
+ CONSOLE     000A      3K    .1%
+ TSO         0034      1K    .0%
+ RRS         0020     272    .0%
+ AUTONETV    0023     136    .0%
+ IBMUSER     0031     136    .0%
+ TCPIP       0035     136    .0%
+ APPC        003B     136    .0%
+ CANSM2RC    004C     136    .0%
+ CANSM2      004D     136    .0%
+ CANSM2EZ    0052     136    .0%
+ CANSETE     003F     112    .0%
+ CANSM2HD    0050      40    .0%
*----------------------------- Bottom of Data -----------------------------*
```

Some OMEGAMON command parameters may actually contain several parameters. For example, suppose you want to display the current exception thresholds and attributes defined for the SWPC and WAIT exceptions. The OMEGAMON XACB major command will display all attributes for all exceptions or a subset of exceptions if you use the LIST parameter. Since the LIST parameter can contain multiple entries (a list of exception IDs) you will need to use *quotation marks* (single or double) or *parentheses* within the INGOMX command:

INGOMX EXECUTE CMD=XACB **PARM='LIST=SWPC WAIT'**
NAME=OMIIMVSA

The XACB response should look similar to:


```
CNMKWIND OUTPUT FROM  INGOMX EXECUTE CMD=XACB PARM='LIST=SWP  LINE 0 OF 19
*------------------------------ Top of Data -------------------------------*
 XACBLIST=SWPC WAIT
: SWPC
+       DISPLAY Parameters:    THRESHOLD Parameters:    XLF Parameters:
:         State=ON              Threshold=5           NOT ELIGIBLE FOR XLF
:         Group=SR              Display=CLR2
:         Bell=OFF              Attribute=NONE
+       BOX Parameters:        CYCLE Parameters:
:         Boxchar=NO BOX        ExNcyc=0
:         Boxclr=NONE           Stop=0 (11402)
:         Boxattr=NONE          Cumulative=11402         >03/15/06 14:52:52<
: WAIT
+       DISPLAY Parameters:    THRESHOLD Parameters:    XLF Parameters:
:         State=ON              Threshold=10          NOT ELIGIBLE FOR XLF
:         Group=OP              Display=CLR3
:         Bell=OFF              Attribute=NONE
+       BOX Parameters:        CYCLE Parameters:
:         Boxchar=NO BOX        ExNcyc=0
:         Boxclr=NONE           Stop=0 (26265)
:         Boxattr=NONE          Cumulative=26265         >03/15/06 14:52:52<
*------------------------------ Bottom of Data ----------------------------*
```

You can also use the XACB command to determine the threshold setting for an exception. Notice that the SWPC threshold is five and the WAIT threshold is ten.

What do you do when you need to pass multiple parameters? For example, LIST and TERSE are two parameters of the OMEGAMON for MVS XACB major command. To pass both parameters using INGOMX:

INGOMX EXECUTE CMD=XACB **PARM=(LIST=SWPC WAIT TERSE)** NAME=OMIIMVSA

This should yield a response similar to:

: SWPC Threshold=5 Display=CLR2 State=ON Bell=OFF
: WAIT Threshold=10 Display=CLR3 State=ON Bell=OFF

The entire parameter string specified by PARM= is passed to OMEGAMON. OMEGAMON will understand, in this example, that you want to display the SWPC and WAIT exception attributes in a compact (terse) format.

5 Complex Automation

You can define more complex automation of OMEGAMON exceptions by using the MESSAGES/USER DATA policy item for your monitor resources. For example:

- Define PASSes to issue different commands to be issued at each PASS as a means of escalation if the exception continues.
- Define CODEs to issue different commands based on the content of the exception message.

PASSes and CODEs are mutually exclusive.

5.1 PASSes

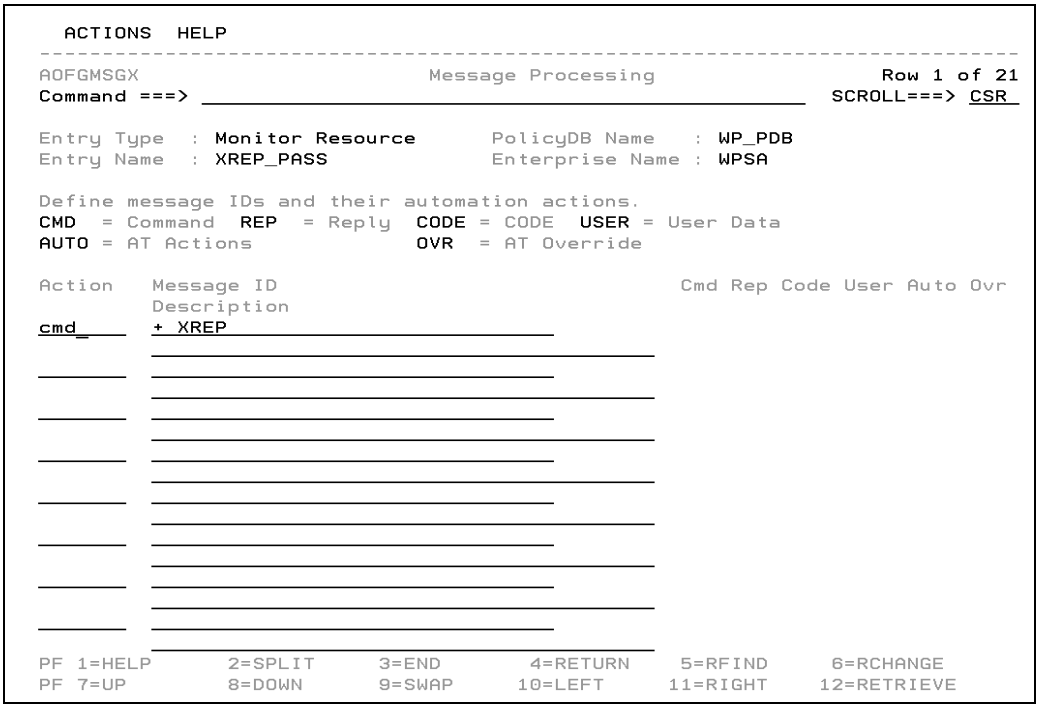
PASSes provide you with a mechanism for taking different actions to resolve an exception. Each pass allows you to issue a different command. One possible implementation of PASSes is to define a series of commands which take more severe actions. You can define up to 99 passes with the counter incremented each time the exception is detected.

For example, suppose you want to monitor for OMEGAMON for MVS XREP exceptions and take more severe actions over a period of time. This example will use a monitor resource called XREP_PASS. The monitor routine will be:

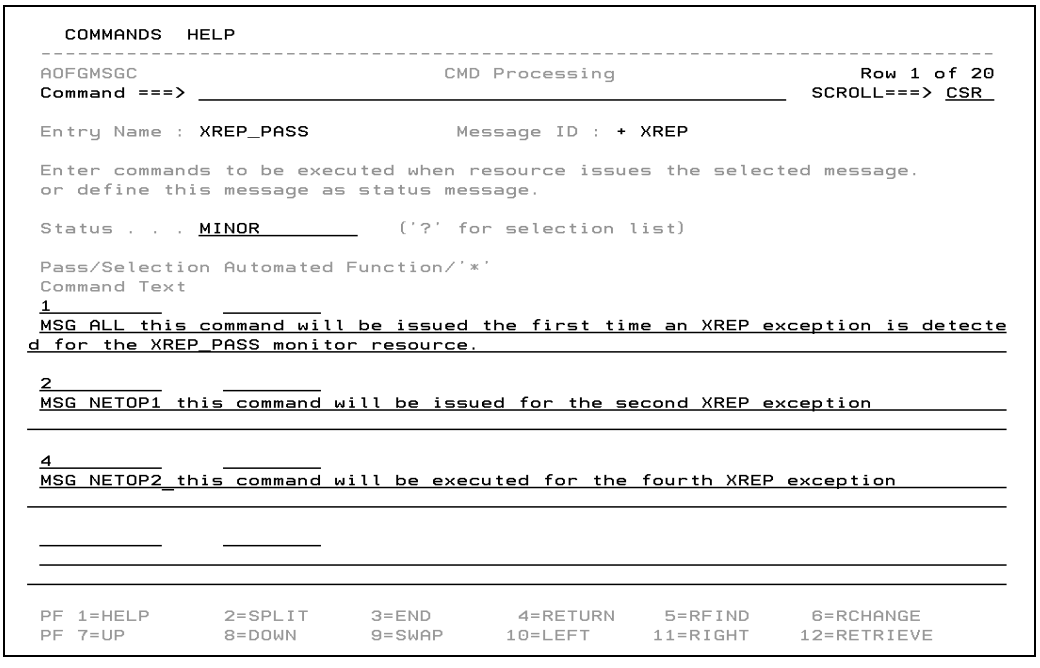
INGMTRAP NAME=OMIIMVSA XTYPE=**XREP**

5.1.1 Define PASSes and Commands

The MESSAGES/USER DATA policy item is used to define the PASSes and commands:



On the *Message Processing* panel (AOFMSGX) define the *Message ID* as **+ XREP** and choose an action of **CMD** to define the PASSes and commands. That will display the *CMD Processing* panel (AOFGMSGC) where you can define the PASSes and their commands:



For this example, PASSes are defined to:

- Issue a MSG ALL command when the first XREP exception is detected.
- Issue a MSG NETOP1 command when the second XREP exception is detected.
- Do nothing when the third XREP exception is detected.
- Issue a MSG NETOP2 command when the fourth XREP exception is detected.

This means that NETOP1 would receive messages from PASS1 and PASS2:

```
NetView V5R2 SA V3        Tivoli NetView   AOFDA NETOP1   03/16/06 15:03:06 A
M AOFDA    DSI039I MSG FROM AUTWRK05 : THIS COMMAND WILL BE ISSUED THE FIRST
           TIME AN XREP EXCEPTION IS DETECTED FOR THE XREP_PASS MONITOR
           RESOURCE.
M AOFDA    DSI039I MSG FROM AUTWRK05 : THIS COMMAND WILL BE ISSUED FOR THE
           SECOND XREP EXCEPTION
```

NETOP2 would receive messages from PASS1 and PASS4:

```
NetView V5R2 SA V3        Tivoli NetView   AOFDA NETOP2   03/16/06 15:05:07
M AOFDA    DSI039I MSG FROM AUTWRK05 : THIS COMMAND WILL BE ISSUED THE FIRST
           TIME AN XREP EXCEPTION IS DETECTED FOR THE XREP_PASS MONITOR
           RESOURCE.
M AOFDA    DSI039I MSG FROM AUTWRK05 : THIS COMMAND WILL BE EXECUTED FOR THE
           FOURTH XREP EXCEPTION
```

Be careful when attempting to generate a *clearing event*. Remember, if you use INGMON CLEARING=YES the PASS count is reset.

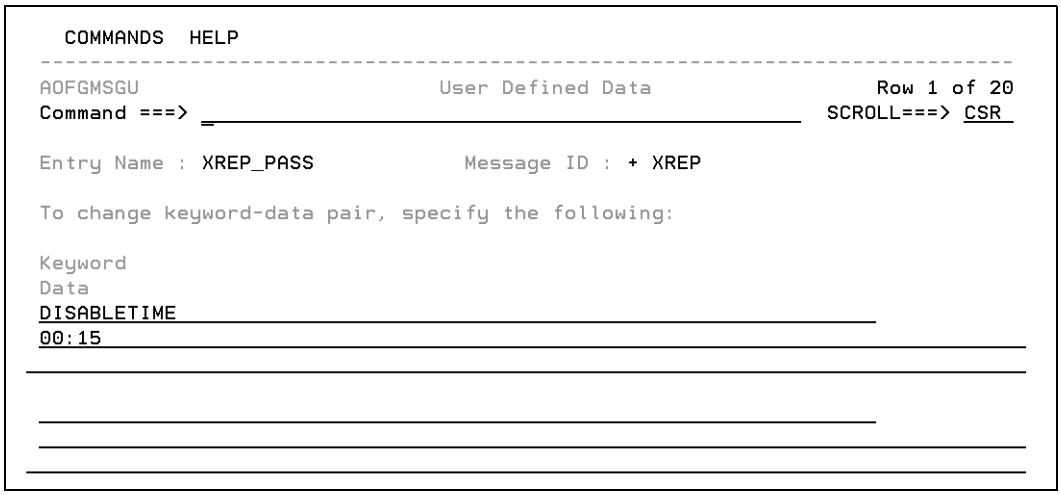
You may also want to define a command to be executed every time an XREP exception is detected. Defining a command to run for every XREP exception was discussed earlier in *3.2 Monitor Resource (MTR) Policy Definitions*.

5.1.2 Temporary Suspend Monitoring

In some cases the automation procedures you define may still be running when the next monitor interval expires. That will cause the monitor routine to be driven again and possibly generate the exception again. Meanwhile your automation procedure is attempting to correct the problem.

You can define a special keyword, DISABLETIME, that will suspend the monitoring. This allows your automation procedure to handle the problem. Monitoring will resume when the DISABLETIME interval expires or when an INGMON CLEARING=YES is issued.

To define a DISABLETIME of 15 minutes, for example, enter an action of **USER** on the *Message Processing* panel. That will display the *User Defined Data* panel (AOFGMSGU):



Define the *Keyword* and *Data* fields as shown and press **PF3** to save. The keyword is the variable name, DISABLETIME, and the data is the value of the variable, 00:15.

This will disable monitoring while the PASS commands attempt to correct the problem.

Note: If you define DISABLETIME for an OMEGAMON exception that generates more than one ING080I message, the first ING080I message will be processed and automation for the subsequent messages will be suspended due to the DISABLETIME definition.

5.1.3 DISPMTR Details for XREP_PASS

When you define PASSes and commands you will also notice that the DISPMTR details panel will display them:

```
INGKYMO1                  SA z/OS  - Command Dialogs      Line  1     of 49
Domain ID   = AOFDA     ---------- DISPMTR  ----------    Date = 03/16/06
Operator ID = NETOP2          Sysplex = SYSPLEX1          Time = 15:02:09

 Monitor          : XREP_PASS/MTR/TIVED1
 System           : TIVED1
 Description      : Define PASSes for XREP exception

 Commands...
    Activate      :
    Deactivate    :
    Monitoring    : INGMTRAP NAME=OMIIMVSA XTYPE=XREP

 Interval         : 00:01

 Monitor Status   : ACTIVE  at  2006-03-16 15:02:06
 Health Status    : MINOR
                    ING080I XREP_PASS/MTR/TIVED1 OMIIMVSA OMIIMVS XREP Number
                    of Outstanding Replies = 6

 Policy Definitions for XREP_PASS ...

    XREP :
      CMD=(PASS1,,'MSG ALL THIS COMMAND WILL BE ISSUED THE FIRST TIME AN XREP
        EXCEPTION IS DETECTED FOR THE XREP_PASS MONITOR RESOURCE.')
      CMD=(PASS2,,'MSG NETOP1 THIS COMMAND WILL BE ISSUED FOR THE SECOND XREP
        EXCEPTION')
      CMD=(PASS4,,'MSG NETOP2 THIS COMMAND WILL BE EXECUTED FOR THE FOURTH XRE

Command ===> _
 PF1=Help     PF2=End      PF3=Return                          PF6=Roll
              PF8=Forward  PF9=Refresh                       PF12=Retrieve
```

Without knowing the exact policy definitions you can clearly see that there are three PASSes defined (PASS1, PASS2, and PASS4) as well as the commands for each PASS.

5.2 CODEs

Code matching provides you with a mechanism for interrogating the OMEGAMON exception message and taking different actions based on the contents of the message. This could be used to provide some complex threshold analysis or to provide Application-specific automation, for example.

When defining CODEs you will need to use the MESSAGES/USER DATA policy item to define:

- CODEs and the values to return to SA z/OS processing.
- Commands to execute based on the CODE values.
- Automation Table overrides to parse the exception message, set the Health Status of the monitor resource, pass the CODE value to the INGMON command.

Suppose you were asked to monitor for OMEGAMON for MVS fixed frame (FXFR) exceptions. This example will use a monitor resource called FIXED_FRAME. The monitor routine will be:

INGMTRAP NAME=OMIIMVSA XTYPE=**FXFR**

Before you begin to define CODEs you should become familiar with the format of the exception message.

You will need to understand how to parse the resulting ING080I message that SA z/OS generates for an exception so that you can properly define an Automation Table override. For example:

```
ING080I FIXED_FRAME/MTR/TIVED1 OMIIMVSA OMIIMVS FXFR STC TSO
| Fixed Frames in use = 49
```

There are 14 tokens in this ING080I message:

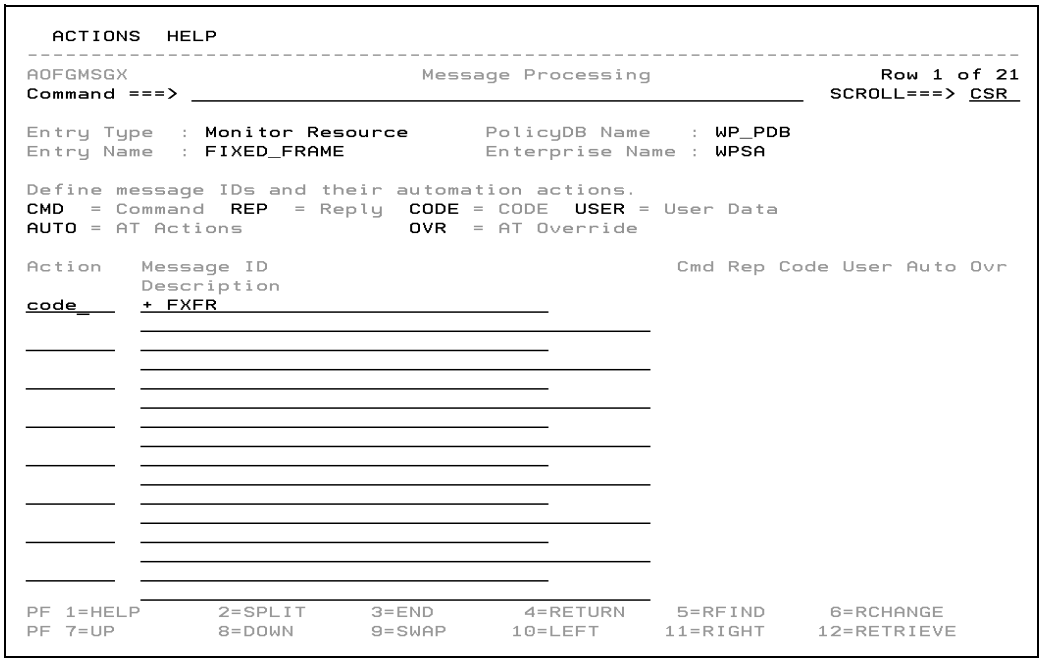
Token Number	Description	Token Value
1	Message ID	ING080I
2	SA z/OS MTR resource name	FIXED_FRAME/MTR/TIVED1
3	Name of session with the OMEGAMON monitor	OMIIMVSA
4	Type of OMEGAMON monitor (OMEGAMON II for MVS)	OMIIMVS
5	Four character exception ID	FXFR
6		STC
7		TSO
8		|
9		Fixed
10		Frames
11		in
12		use
13		=
14		49

The first five tokens will always be the message ID, MTR resource name, session name, session type, and exception ID. The remaining tokens are the exception message from the OMEGAMON monitor.

This example will define CODEs using the value of the seventh token (the *STC name*) that can then drive differing commands for each Application based on the *STC name*. To keep this discussion easy to follow assume that you only need the seventh token which means you will only need one set of CODE definitions. SA z/OS supports three CODE definitions.

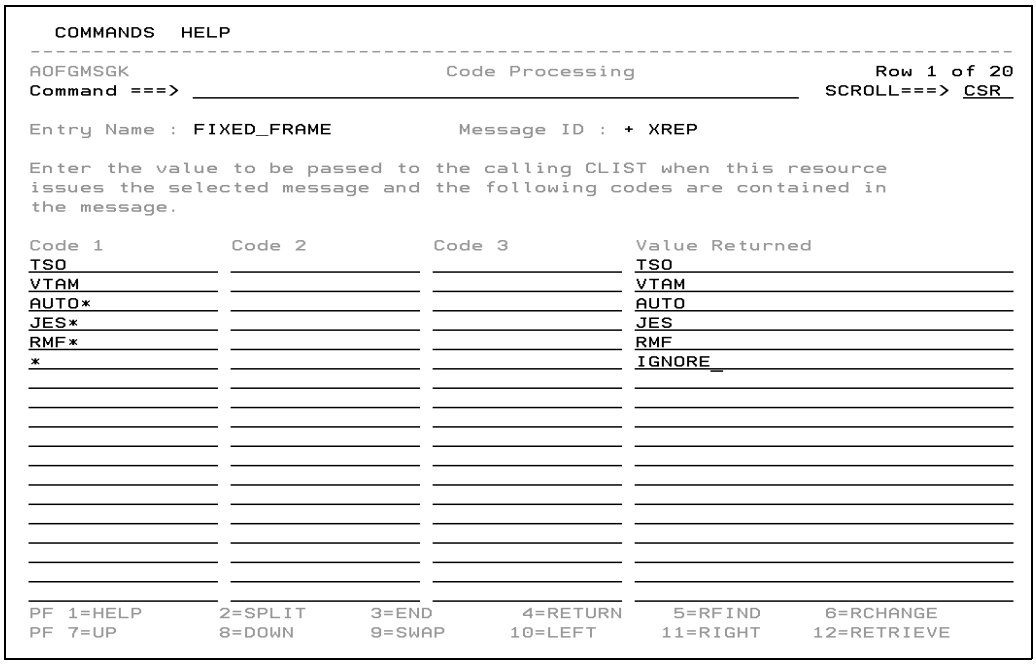
5.2.1 Define CODEs and Values

Begin by selecting the MESSAGES/USER DATA policy item to define the CODEs and commands:



This is the *Message Processing* panel (AOFMSGX) for a monitor resource to monitor for FXFR exceptions from OMEGAMON for MVS.

Define the *Message ID* as **+ FXFR** and choose an action of **CODE** to define the CODEs and their values. That will display the *Code Processing* panel (AOFGMSGK) where you can define the CODEs and their values:



For this example, when an FXFR exception is detected automation will use these values for CODE1:

- When the STC token is TSO return a value of TSO.
- When the STC token is VTAM return a value of VTAM.
- When the STC token begins with AUTO return a value of AUTO. This will handle all AUTO* Applications (AUTONETV, AUTOSSI, AUTOMGR, for example).
- When the STC token begins with JES return a value of JES. This will handle all JES* Applications (JES2, JES2AUX, JES2MON, for example).
- When the STC token begins with RMF return a value of RMF. This will handle all RMF* Applications (RMF, RMFGAT, for example).
- Otherwise return a value of IGNORE. This will use INGMON to set the Health Status of the monitor resource but will not perform any additional automation.

Press **PF3** to save the CODE definitions.

This example defines six values to be used for CODE1.

You will use pass CODE1 to the INGMON routine when you create the Automation Table override.

5.2.2 Define Commands

The next step in this process involves defining commands for each of the CODE1 values. Enter **CMD** for the action on the *Message Processing* panel to display the *CMD Processing* panel.

Similar to the commands for PASSes, you will define a command to be driven for each of the six CODE1 values defined. The name of the *Pass/Selection* field will match the value returned from the CODE1 code matching.

```
  COMMANDS  HELP
------------------------------------------------------------------------------
AOFGMSGC                          CMD Processing                    Row 1 of 20
Command ===> ____________________________________________________  SCROLL===> CSR

Entry Name : FIXED_FRAME           Message ID : + FXFR

Enter commands to be executed when resource issues the selected message.
or define this message as status message.

Status . . . _____________    ('?' for selection list)

Pass/Selection Automated Function/'*'
Command Text
JES_______   ________
MSG NETOP1 fixed frame exception for 1 of the JES APLs

AUTO______   ________
MSG NETOP1 fixed frame exception for 1 of the AUTO APLs

VTAM______   ________
MSG NETOP1 fixed frame exception for VTAM

TSO_______   ________
MSG NETOP1 fixed frame exception for TSO_

PF 1=HELP      2=SPLIT     3=END       4=RETURN    5=RFIND     6=RCHANGE
PF 7=UP        8=DOWN      9=SWAP     10=LEFT     11=RIGHT    12=RETRIEVE
```

This example *CMD Processing* panel defines a different MSG NETOP1 to be issued based on the CODE1 value returned from the code matching that was defined. In most cases, you will probably code a REXX EXEC to be called to perform a series of commands. The MSG NETOP1 is used for illustrative purposes only.

This panel shows four of the six commands. The two commands not shown are:

- The command for CODE1=IGNORE: MSG NETOP1 fixed frame exception ignored.
- The command for CODE1=RMF: MSG NETOP1 fixed frame exception for one of the RMF APLs.

Press **PF3** to save your command definitions.

You have defined the CODE values and commands. The next step is to define an Automation Table override to call INGMON with the appropriate parameters to execute the commands that you defined for each of the Applications (token seven of the ING080I message).

5.2.3 Define Automation Table Overrides

Next, define an Automation Table override to call INGMON to set the Health Status and call the CODE1 commands that are defined.

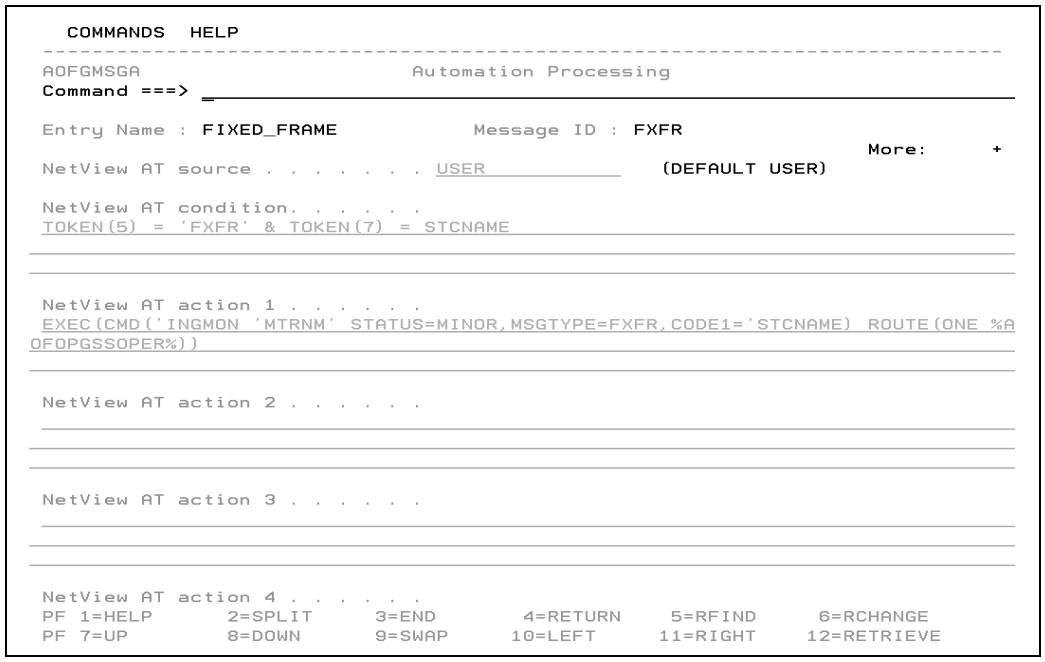
On the *Message Processing* panel, enter **OVR** for the action to display the *Automation Processing* panel (AOFMSGA). Initially this panel will display the default Automation Table entry for the FIXED_FRAME monitor resource for an ING080I message:

```
  COMMANDS  HELP
------------------------------------------------------------------------------
AOFGMSGA                      Automation Processing
Command ===> _

Entry Name : FIXED_FRAME           Message ID : FXFR
                                                          More:      +
NetView AT source . . . . . . . DEFAULT          (DEFAULT USER)

NetView AT condition. . . . . .
TOKEN(5) = 'FXFR'


NetView AT action 1 . . . . . .
EXEC(CMD('INGMON 'MTRNM)ROUTE(ONE %AOFOPGSSOPER%))


NetView AT action 2 . . . . . .


NetView AT action 3 . . . . . .


NetView AT action 4 . . . . . .
PF 1=HELP      2=SPLIT     3=END       4=RETURN    5=RFIND     6=RCHANGE
PF 7=UP        8=DOWN      9=SWAP     10=LEFT     11=RIGHT    12=RETRIEVE
```

When an ING080I message is generated by SA z/OS for an OMEGAMON exception and the fifth token of the message is FXFR then call INGMON with the monitor resource name. If you recall, the fifth token will always be the exception ID. In this case it is FXFR.

You need to modify:

- **NetView AT condition**: Parse the seventh token and save into a user variable, STCNAME. The variable will be passed to INGMON as the value for CODE1.
- **NetView AT action 1**: Modify the INGMON command.
  - Add STATUS= with a valid Health Status (MINOR).
  - (Optional) Add MSGTYPE= with the exception ID (FXFR).
  - Add CODE1= Using the variable for token seven (STCNAME).
- **NetView AT action 2**: You can define additional automation actions to take if an ING080I is received for FXFR exceptions.



These changes create an Automation Table override. The color of the fields changes so that you can clearly understand that you are modifying the Automation Table statements used by SA z/OS.

An example ING080I message for a fixed frame exception is:

```
ING080I FIXED_FRAME/MTR/TIVED1 OMIIMVSA OMIIMVS FXFR STC TSO
| Fixed Frames in use = 49
```

The ING080I message will drive the Automation Table to call INGMON as defined with the AT override:

```
INGMON FIXED_FRAME/MTR/TIVED1,STATUS=MINOR,MSGTYPE=FXFR,CODE1=TSO
```

If you are not familiar with NetView Automation Table coding see the *IBM Tivoli NetView for z/OS Automation Guide*.

Press **PF3** to save the Automation Table overrides.

5.2.4 CODEs Summary

When you are done defining the CODEs, values, commands, and so on your MESSAGES/ USER DATA policy item should look similar to:


```
  ACTIONS  HELP
------------------------------------------------------------------------------
AOFGMSGX                        Message Processing                  Row 1 of 21
Command ===> _                                                  SCROLL===> CSR

Entry Type  : Monitor Resource      PolicyDB Name   : WP_PDB
Entry Name  : FIXED_FRAME           Enterprise Name : WPSA

Define message IDs and their automation actions.
CMD  = Command  REP  = Reply  CODE = CODE  USER = User Data
AUTO = AT Actions             OVR  = AT Override

Action   Message ID                                  Cmd Rep Code User Auto Ovr
         Description
_______  + FXFR                                      6       6              *
         ____________________________________________
_______  ____________________________________________
         ____________________________________________
```

The *Message Processing* panel (AOFGMSGX) tells you that there are six commands, six CODEs, and an Automation Table override defined.

5.2.5 DISPMTR for FIXED_FRAME

The DISPMTR details panel for the FIXED_FRAME monitor resource will look similar to:

```
INGKYMO1                  SA z/OS  - Command Dialogs      Line  11    of 109
Domain ID    = AOFDA     ---------- DISPMTR  ----------    Date = 03/17/06
Operator ID = NETOP1          Sysplex = SYSPLEX1          Time = 13:04:12


 Monitor Status  : ACTIVE  at  2006-03-17 12:57:04
 Health Status   : MINOR
                   ING080I FIXED_FRAME/MTR/TIVED1 OMIIMVSA OMIIMVS FXFR STC
                   APPC | Fixed Frames in use = 217

 Policy Definitions for FIXED_FRAME ...

    FXFR :
      CMD=(JES,,'MSG NETOP1 FIXED FRAME EXCEPTION FOR 1 OF THE JES APLS')
      CMD=(AUTO,,'MSG NETOP1 FIXED FRAME EXCEPTION FOR 1 OF THE AUTO APLS')
      CMD=(VTAM,,'MSG NETOP1 FIXED FRAME EXCEPTION FOR VTAM')
      CMD=(TSO,,'MSG NETOP1 FIXED FRAME EXCEPTION FOR TSO')
      CMD=(IGNORE,,'MSG NETOP1 FIXED FRAME EXCEPTION IGNORED')
      CMD=(RMF,,'MSG NETOP1 FIXED FRAME EXCEPTION FOR 1 OF THE RMF APLS')
      CODE=(TSO,,,"TSO")
      CODE=(VTAM,,,"VTAM")
      CODE=(AUTO*,,,"AUTO")
      CODE=(JES*,,,"JES")
      CODE=(RMF*,,,"RMF")
      CODE=(*,,,"IGNORE")
      DISABLETIME=00:15

 History (maximum is 25) ...

Command ===> _
 PF1=Help      PF2=End      PF3=Return                          PF6=Roll
 PF7=Back      PF8=Forward  PF9=Refresh                        PF12=Retrieve
```

DISPMTR details displays the policy for the FXFR exception that is defined. The six commands and six CODEs are shown plus the DISABLETIME user keyword. The Health Status of the monitor resource is MINOR. The last ING080I is also shown.

If you browse the Automation Table you will see the override that is defined:

```
IF
MSGID = 'ING080I' & TOKEN(2) = MTRNM
 THEN BEGIN;
*
*
IF
TOKEN(5) = 'FXFR' & TOKEN(7) = STCNAME
 THEN
 EXEC(CMD('INGMON 'MTRNM'
STATUS=MINOR,MSGTYPE=FXFR,CODE1='STCNAME) ROUTE(ONE
%AOFOPGSSOPER%));
```

The first IF statement checks for message ING080I and parses the second token into the MTRNM variable. The second IF statement checks if the fifth token of the ING080I message is FXFR and parses the STC task name into the STCNAME variable. INGMON is then called to set the Health Status to MINOR and pass the STCNAME in CODE1.

6 Security

Your installation may require security such as who can log on to the OMEGAMON monitor Applications and which OMEGAMON commands they can issue. Passwords for the sessions between the SA z/OS Automation Agent and each OMEGAMON monitor Application can also be secured.

The security can be defined within OMEGAMON or NetView or both.

6.1 OMEGAMON Security

OMEGAMON supports security with SAF products such as RACF or an internal security table with OMEGAMON. Items that can be secured are user IDs, passwords, commands, and command parameters.

The best approach is to define the user ID to OMEGAMON:

- Internally in the OMEGAMON security table or externally to a SAF product.
- Must be able to access an INITIAL*n* profile to connect and issue commands.
- Should be granted the highest level of security defined for the installation.

Then, use NetView security to control which NetView operators (including automation tasks) have access to the sessions and commands.

6.2 NetView Security

Using the NetView Command Authorization Table (CAT) you can define security to control access to the INGSESS and INGOMX commands and parameters. These two commands control who can start or stop a session and which commands they can issue. For example, a subset of operators should be granted access to INGSESS to start or stop the sessions between SA z/OS and the OMEGAMON monitors. You should also use the CAT table to control access to INGOMX and even the OMEGAMON commands such as CSAA or KILL.

The two SA z/OS commands you will need to code CAT table statements for are:

- INGOMX: Use INGROMX0 as the name in the CAT table

- INGSESS: Use INGRYSS0 as the name in the CAT table

To properly define command security you will need to familiarize yourself with the syntax of the INGOMX and INGSESS commands and their parameters. The command syntax is not discussed in this document. You can use the NetView HELP command to display the syntax and description of the INGOMX and INGSESS commands.

In general, you will need to define CAT table entries to:

- Restrict access to the INGOMX and INGSESS commands and their parameters.
- Define operators (including automation tasks) to one or more groups.
- Grant group access to the INGOMX and INGSESS commands and their parameters.

6.2.1 Restricting Access to INGOMX and INGSESS Commands and Parameters

This section deals with restricting access to the commands and their parameters.

Define a PROTECT statement to restrict INGOMX access to **all** sessions:

```
PROTECT *.*.INGROMX0.NAME.*
```

Define a PROTECT statement to restrict INGOMX access to a session called OMIIMVSA:

```
PROTECT *.*.INGROMX0.NAME.OMIIMVSA
```

Define a PROTECT statement to restrict INGOMX access to **all** OMEGAMON commands:

```
PROTECT *.*.INGROMX0.CMD.*
```

Define PROTECT statements to restrict access to INGSESS NAME=, REQ=START, and REQ=STOP:

```
PROTECT *.*.INGRYSS0.NAME.*
PROTECT *.*.INGRYSS0.REQ.START
PROTECT *.*.INGRYSS0.REQ.STOP
```

By default, this will permit all users to issue an INGSESS REQ=DISPLAY to display all sessions and INGSESS REQ=DETAIL to display session details for a specific session if the user has been granted access to the session itself (NAME= parameter).

6.2.2 Define Two Operator Groups

Define at least two groups of operators. The first group will be strictly operators and they will not be given access to start or stop sessions, for example. The second group will be administrators who will be given access to the same functions as the operators plus functions such as the start or stop of sessions.

Define the operators group:

```
GROUP OMOPERS OPER1,OPER2,ADMIN1,ADMIN2
```

Define the administrators group:

```
GROUP OMADMINS ADMIN1,ADMIN2
```

Note: You should define all SA z/OS work autotasks (AUTWRK*xx*) and the AUTRPC autotask in the OMADMIN group. Optionally, you may choose to define a third group of operators for the autotasks.

6.2.3 Grant Access to INGOMX and INGSESS Commands and Parameters

Define several PERMIT statements for each group of operators to grant access to INGOMX, INGSESS, and their respective command parameters.

Define a PERMIT statement to allow operators in the OMOPERS group access to the OMIIMVSA session when using the INGOMX command (INGOMX NAME=OMIIMVSA):

```
PERMIT OMOPERS *.*.INGROMX0.NAME.OMIIMVSA
```

Define a PERMIT statement to allow the operators in the OMADMINS group access to the OMEGAMON KILL command (INGOMX EX NAME=OMIIMVSA CMD=KILL):

```
PERMIT OMADMINS *.*.INGROMX0.CMD.KILL
```

Define other PROTECT and PERMIT statements as needed to authorize access to any other OMEGAMON commands:

```
PROTECT *.*.INGROMX0.CMD.OM_cmd_name
PROTECT *.*.INGROMX0.NAME.session_name
PERMIT OperGroup *.*.INGROMX0.NAME.session_name
PERMIT OperGroup *.*.INGROMX0.CMD.OM_cmd_name
```

In general, protect the session with the NAME parameter and the command with the CMD parameter. Then, permit the operators in *OperGroup* access to the NAME and CMD parameters.

For more information on SA z/OS security see the *IBM Tivoli System Automation for z/OS: Planning and Installation* manual.

For more information on NetView command security see the *IBM Tivoli NetView for z/OS Security Reference* manual.

6.2.4 Authorizing TRAP Commands

Retrieving exceptions from an OMEGAMON monitor Application requires that autotasks be granted access to INGOMX TRAP.

Suppose, for example, you do not want operators issuing INGOMX TRAP commands. You will need to add a PROTECT statement to your CAT table to restrict access to an *internal command* used by SA z/OS to retrieve exceptions from an OMEGAMON monitor. The name of the internal command is **EXSY**.

Note: The EXSY command is used by the OMEGAMON for MVS, DB2, and CICS monitors. OMEGAMON for IMS uses a different command, **XIMS**. You can secure the XIMS command the same as the EXSY command.

EXSY will be issued when an INGOMX TRAP,XTYPE=*xxxx*,NAME=*session_name* command is issued. To restrict access:

```
PROTECT *.*.INGROMX0.CMD.EXSY
```

Then, add PERMIT statements as necessary to allow the tasks access to the EXSY command:

```
PERMIT OMADMINS *.*.INGROMX0.CMD.EXSY
```

You will need to permit these tasks to use EXSY:

- SA z/OS work autotasks (AUTWRK*xx*).
- AUTRPC autotask.
- Any NetView operators that need to retrieve exceptions.

If an operator attempts to retrieve an exception (for example, INGOMX TRAP,XTYPE=XREP,NAME=OMIIMVSA) they should see messages similar to:

```
BNH236E 'OPER1' IS NOT AUTHORIZED TO USE THE KEYWORD 'CMD'
AND VALUE 'EXSY' COMBINATION
BNH237E THE KEYWORD 'CMD' AND VALUE 'EXSY' ARE PROTECTED BY
COMMAND IDENTIFIER '*.*.INGROMX0.CMD.EXSY' IN
'TBLNAME=OMSEC'
```

This will inform the operators that they are not authorized to issue the EXSY command.

6.2.5 Handling Special Commands

Some OMEGAMON commands contain characters that require special handling in the CAT table definitions. The **.RMF** command is an example. When you define a PROTECT or PERMIT statement for .RMF you need to replace the period with an *at sign* (@):

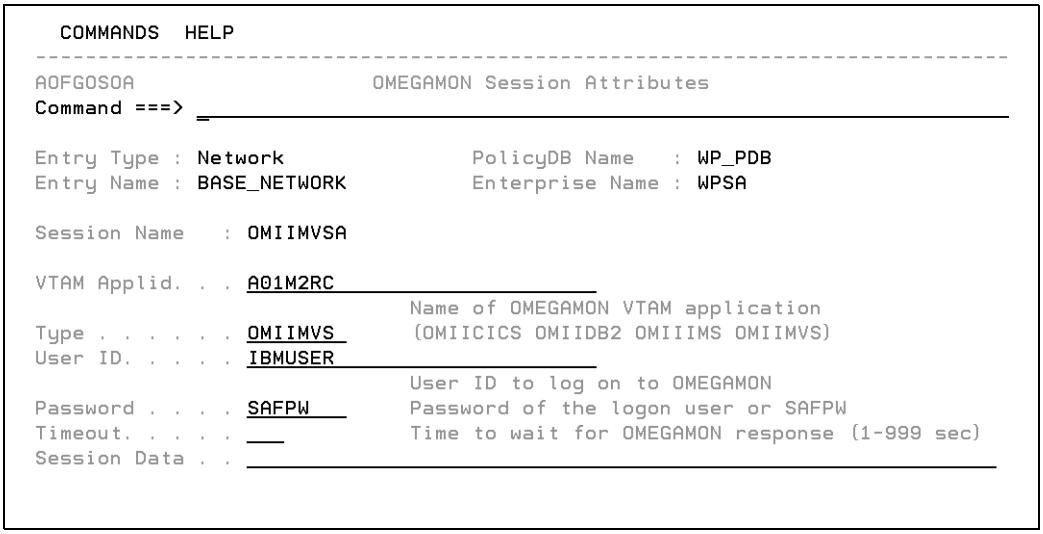
```
PROTECT *.*.INGROMX0.CMD.@RMF
PERMIT OMADMINS *.*.INGROMX0.CMD.@RMF
```

6.3 Password Security

You can define the session password in two ways:

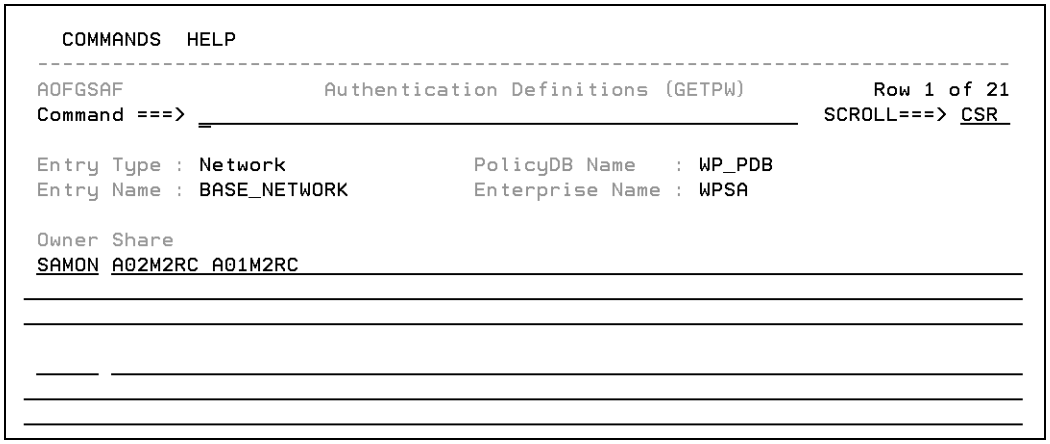
- As part of the OMEGAMON SESSIONS policy item when you define the session between SA z/OS and an OMEGAMON monitor. The examples shown in this document define the session password this way.
  - Using the OMEGAMON SESSIONS policy item may not be secure enough for your environment since the password can be seen by anyone who has access to the Customization Dialog.
- Alternatively, you can define the password field in the OMEGAMON SESSIONS policy item as **SAFPW**. SAFPW forces SA z/OS to use encrypted passwords in the NetView password data set, EZLPSWD.

For example, to define SAFPW for a session called OMIIMVSA, begin with option **39** (NTW, Network) and **s**elect the OMIIMVSA session within the OMEGAMON SESSIONS policy item:



The *Password* field is set to **SAFPW**, meaning that for this session, OMIIMVSA, SA z/OS will use the NetView password data set, EZLPSWD, and the GETPW command to manage the passwords.

If you specify SAFPW for the session password you must define the AUTHENTICATION policy item:



This defines an owner of SAMON and a share list of A02M2RC and A01M2RC where:

- **Owner**: Specifies the name used as domain name parameter for the NetView GETPW command. In this example, the domain parameter will be **SAMON**.
- **Share**: Is a list of all the APPL IDs that use the owner to access the appropriate entry in the NetView password data set. The APPL IDs may delimited by blank or comma. There are two APPL IDs in this example, the two OMEGAMON for MVS APPL IDs.

SA z/OS will use the NetView GETPW command to retrieve the session password.

One advantage of using GETPW and encrypted passwords is that the password values are automatically updated every 30 days by the GETPW command.

You must use the GETPW command to define the password values initially. For example:

```
GETPW operID SAMON,INIT=initial_pswd,MASK=%A%N%N%A%A%A%A%A
```

The parameters for the GETPW command are:

- *operID* is the OMEGAMON user ID (for example, IBMUSER in this document).
- SAMON is the owner for the password data set.
- *initial_pswd* is the initial value of the password using the format specified by the MASK parameter.

- MASK defines the format of the password value. For example a string of eight characters, beginning with a letter, followed by two numbers and then five characters.

All users (operators and autotasks) of a session between SA z/OS and an OMEGAMON monitor will need to be authorized to use the GETPW command. For example, you will need to define additional PROTECT and PERMIT statements in your CAT table for the GETPW command.

The NetView password data set requires additional customization when you install NetView. See the *IBM Tivoli NetView Installation: Configuring Additional Components* for more details.

Conclusion

Summary

With SA z/OS 3.1 and the OMEGAMON *Classic* monitors for MVS, CICS, DB2, and IMS you have a very powerful base for automation. The OMEGAMON monitors collect data about the Applications and zSeries resources. SA z/OS can access that data and determine the root cause of a problem more quickly to take corrective actions more quickly.

This document has shown you how to customize your policy to define sessions that connect SA z/OS with the OMEGAMON monitors. Using the INGSESS command you can manage those connections (start, stop, display status, display statistics).

Using SA z/OS monitor resources you can retrieve OMEGAMON exceptions and use them to:

- Set the Health Status of SA z/OS resources.
- Drive your own automation routines to take corrective actions.

Using the INGOMX command you have seen several examples of how you can issue OMEGAMON commands from the SA z/OS Automation Agent NetView. INGOMX can also be used to retrieve OMEGAMON exceptions.

The document has also shown you how to define SA z/OS PASSes and CODEs for more robust automation.

This document has also provided you with valuable information in terms of securing SA z/OS commands, parameters, and session passwords. Examples were provided of each to assist you with implementing security in your environment more rapidly.

Acknowledgements

A special *Thank You* is extended to Walter Schueppen, Hans Ulrich Geissler, Juergen Holtz, and the SA z/OS Development Team for their reviews of this document to ensure its accuracy and content.

Resources

IBM Tivoli System Automation for z/OS 2.3: A Primer to Monitor Resources: This White Paper provides valuable information on defining and using SA z/OS monitor resources. The information is written for the SA z/OS 2.3 release, but applies to SA z/OS 3.1 as well.

OMEGAMON for MVS Reference Manual: Contains information related to OMEGAMON for MVS exceptions (XREP, SWPC, FXFR, and so on).

OMEGAMON II for MVS Command Language Reference Manual: Contains information related to OMEGAMON major, minor, and immediate commands.

IBM Tivoli NetView for z/OS Programming: PIPEs: Contains information related to writing procedures that use NetView PIPEs.

IBM Tivoli NetView for z/OS Automation Guide: Contains detailed information that may be needed by the System Administrator when defining Automation Table overrides in the SA z/OS MESSAGES/USER DATA policy items.

IBM Tivoli NetView for z/OS Security Reference manual: This book should be used if you need to control access to commands (SA z/OS and OMEGAMON) and parameters.

IBM Tivoli NetView Installation: Configuring Additional Components manual: This book will be needed if you want to use encrypted passwords for the sessions between the SA z/OS Automation Agent and the OMEGAMON monitors.

The WAITSWPC REXX code is provided in Appendix A of this document.

Appendix A: WAITSWPC REXX EXEC

This is the source for the WAITSWPC REXX EXEC used in this document.

```
/* REXX EXEC */
/*************************************************************/
/* */
/* WAITSWPC: */
/* (0) Parse out input parms and set local variables */
/*     Expected input: */
/*     NAME=session_name (ie; OMIIMVSA) */
/* (1) Using INGOMX, retrieve SWPC exceptions */
/* (2) Build list of resources who have SWPC exception against them
*/
/* (3) Issue INGOMX, retrieve WAIT exceptions */
/* (4) Filter WAIT response for only the resources that also have*/
/*     an SWPC exception against them */
/* (5) Issue INGMON to update the Health Status of the monitor */
/* (6) Issue INGSET to update the Health Status of the resources*/
/* (7) End with return_code of 8 (DEFER) if exceptions detected */
/*     Else with a return_code of 3 (NORMAL) */
/*                                                               */
/*                                                               */
/* Notes:                                                        */
/* - Task Global WAITSWPC contains the Applications (APLs) that
have  */
/*   had their Health Status modified by this routine. If you wish
to */
/*  generate a “clearing event” then you should loop through the
*/
/*  Applications in WAITSWPC to also reset their Health Status.
*/
/*                                                               */
/*************************************************************/

TRACE o

/*************************************************************/
/* INITIALIZE LOCAL VARIABLES */

parse arg ‘NAME=’p1 .

If p1 = ‘’ then                                  /* parm error? */
  Return_Code = 2                               /* cannot continue */
Else
  Return_Code = 3                                /* good to go */

loc_str = ‘’
msg_str = ‘’
res_str = ‘’

‘GLOBALV GETT SUBSAPPL’                          /* get MTR name */
```

```
'GLOBALV GETT WAITSWPC'                          /* get APL list */
'GLOBALV GETC AOFSYSNAME'                        /* get sys name */


/*****************************************************************/


If Return_Code = 3 then
  Do                                          /* mainline process */
    Call Issue_SWPC
    If Loc_Str <> '' then /* SWPC exceptions? */
      Do
        Call Issue_WAIT
        Call Update_Status
         Return_code = 8 /* defer health status*/
      End
  End                                         /* mainline process */

'GLOBALV PUTT WAITSWPC'                          /* save list APLs */

Return return_code



Issue_SWPC:
/*****************************************************************/
/* Retrieve SWPC exceptions and build a PIPE LOC string that
contains */
/* the Applications with SWPC. The LOC will be used when WAIT */
/* exceptions are retrieved. */
/*****************************************************************/

'PIPE NETV ',
  'ingomx trap xtype=swpc name='p1,
  ' | CORR',
  ' | COLL',
  ' | Stem SWPC.'

Do i=1 to swpc.0
  /* + SWPC STC SNMPQE      | Excessive Swap counts = 14
*/
  Parse var swpc.i . . . Appl .
  loc_str = loc_str '/'Appl'/'
End

loc_str = Strip(loc_str)

Return


Issue_WAIT:
/*****************************************************************/
/*                                                               */
/*                                                               */
```



```
/*****************************************************************/

'PIPE NETV ',
  'ingomx trap xtype=wait name='p1,
  ' | CORR',
  ' | LOC 'loc_str,
  ' | COLL',
  ' | Stem wait.'

Do i=1 to wait.0
  /* + SWPC STC SNMPQE      | Excessive Swap counts = 14
*/
  Parse var wait.i . . . Appl .
  res_str = res_str' 'Appl

  /* If the APL is not already in the list, add it. */
  /* The list will be needed if a "clearing event" is generated. */
  If POS(Appl,WAITSWPC) = 0 then                /* new APL?  */
    WAITSWPC = WAITSWPC' 'Appl                 /* Add to list    */

End

res_str = Strip(res_str)
WAITSWPC = Strip(WAITSWPC)

Return


Update_Status:
/*****************************************************************/
/*                                                               */
/*                                                               */
/*****************************************************************/

msg_str = "'Correlated exceptions found for: "res_str"'"

/* INGMON to update health of MTR (subsappl)
*/
'INGMON 'subsappl',STATUS=WARNING,MSG='msg_Str

Do i=1 to words(res_str)                       /* set APL Health */
  Resname = word(res_str,i)

  /* INGSET to update health of each resource in the list */
  'INGSET SET 'resname'/APL/'aofsysname',HEALTH=WARNING'
End                                            /* set APL Health */

Return
```

