



IBM Tivoli NetView for z/OS 6.1: Fundamentals

Student's Training Guide

Course: TZ203 ERC: 1.0

August 2011

© Copyright IBM Corp. 2011. All Rights Reserved.

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

IBM, the IBM logo, and ibm.com are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in this publication to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth, savings or other results.

Printed in Ireland

Table of contents

Unit 1: NetView for z/OS basics

Introduction	1-2
Objectives	1-2
Lesson 1: NetView overview	1-3
NetView for z/OS	1-3
NetView program features	1-4
IP management	1-5
Automation introduction	1-6
Sysplex monitoring	1-8
Problem management	1-10
NetView Enterprise Management Agent (EMA)	1-11
Security	1-12
NetView 6.1 information center publications	1-13
Lesson 2: IBM Tivoli NetView packaging and installation	1-14
Packaging and installation of NetView host code	1-14
NetView EMA code: z/OS	1-15
NetView EMA data files: Distributed	1-17
Lesson 3: NetView structure and components	1-18
NetView components	1-18
Command facility	1-19
Hardware monitor	1-20
Session monitor	1-22
Terminal access facility	1-23
SNA Topology Manager (SNATM)	1-24
4700 support facility	1-25
Automated Operations Network (AON)	1-26
MultiSystem Manager	1-27
Browse facility	1-29
Automation table	1-30
Subsystem interface	1-31
Resource Object Data Manager (RODM)	1-32
Graphic Monitor Facility Host Subsystem (GMFHS)	1-33
NetView Enterprise Management Agent (EMA)	1-34
Correlation engine	1-35
Common base event manager	1-36
Event/Automation Service (E/AS)	1-37
Lesson 4: NetView application procedure parameters and data sets	1-38
NetView procedure parameters	1-38
Data definition names	1-40
NetView subsystem interface	1-42
Lesson 5: NetView 3270 interface	1-44
NetView application program	1-44
Providing an operator ID and password	1-45
NetView LISTA DSIPARM command	1-46
Canzlog	1-47
Canzlog	1-48
Student exercise	1-49
Lesson 6: Customizing NetView by using CNMSTYLE	1-50

CNMSTYLE (1 of 2)1-50
CNMSTYLE (2 of 2)1-51
Included members from DSIPARM (1 of 2)1-52
Included members from DSIPARM (2 of 2)1-53
Included member CNMSTUSR from DSIPARM1-54
Included member CxxSTGEN from DSIPARM1-55
CNMSTYLE changes during NetView execution1-56
Data REXX logic1-57
Towers and subtowers (1 of 6)1-58
Towers and subtowers (2 of 6)1-59
Towers and subtowers (3 of 6)1-60
Towers and subtowers (4 of 6)1-61
Towers and subtowers (5 of 6)1-62
Towers and subtowers (6 of 6)1-63
Tower and subtowers for Tivoli Enterprise Monitoring Agent1-64
Commands automatically run after initialization1-65
Setting up security1-66
SECOPTS statements and options in CNMSTYLE1-67
Student exercise1-68
Lesson 7: CNMSTYLE report generator1-69
CNMSTYLE report generator features1-69
CNMSJCRG job1-71
Typical CNMSJCRG job1-72
CNMSTYLE report: First section1-74
%INCLUDE structure1-75
General NetView statements1-76
CNMSTYLE report: Second section1-77
Example of TCP/IP tower functions1-78
CNMSTYLE report: Third section1-79
auxInitCmd statements1-80
CNMSTYLE report: Fourth section1-81
Data REXX statements1-82
Student exercise1-83
Lesson 8: NetView administration1-84
Defining NetView operators1-84
Defining NetView operator profiles (1 of 3)1-85
Defining NetView operator profiles (2 of 3)1-86
Defining NetView operator profiles (3 of 3)1-87
Example of DSIOPF defining operators1-88
Example of a PROFILEN definition1-89
DSIPARM member CNMSCAT21-90
SAF administration with CNMSAF21-91
Lesson 9: NetView IP management1-92
Management of SNA over IP1-92
Dynamic virtual IP address1-93
Support for DVIPA1-94
Dynamic IP resource discovery1-95
IP connection management1-96
Packet trace collection and formatting1-97
OSA trace collection and formatting1-98
Formatted trace1-99
Command support1-100
Automated responses to intrusions1-101
SNMP support for commands1-102
Network address translation1-103
IPv6 support and enablement1-104
Summary1-105

Unit 2: NetView user interfaces and product integration

Introduction	2-2
Objectives	2-2
Lesson 1: NetView web interface	2-3
NetView web application	2-3
Lesson 2: NetView web services gateway	2-4
NetView web services gateway features	2-4
NetView web services gateway flow	2-5
NetView web services gateway example	2-6
Lesson 3: NetView Management Console (NMC) interface	2-7
NetView Management Console (NMC)	2-7
Performing tasks with NMC (1 of 2)	2-8
Performing tasks with NMC (2 of 2)	2-9
Lesson 4: NetView EMA	2-10
NetView EMA overview	2-10
Performing tasks with the NetView EMA	2-11
NetView EMA and data collectors	2-12
Deployment	2-13
Lesson 5: Tivoli Enterprise Portal	2-14
Architecture and components	2-14
Two Tivoli Enterprise Portal modes: Desktop client and browser	2-15
NetView EMA Navigator	2-16
NetView Health workspace	2-18
TCPIP Connection Data workspace	2-19
NetView EMA views	2-20
Online Help	2-21
Summary	2-22
Product-provided situations	2-23
Take Action commands	2-24
Take Action command execution	2-25
NetView Command Response workspace example	2-26
Historical data collection (1 of 3)	2-27
Historical data collection (2 of 3)	2-28
Historical data collection (3 of 3)	2-29
Lesson 6: NetView product integration with IBM Tivoli OMEGAMON XE products	2-30
Integration with IBM Tivoli OMEGAMON XE	2-30
IBM Tivoli OMEGAMON XE cross-product workspace links	2-32
Lesson 7: Tivoli Enterprise Portal security	2-33
User authorization and identification for the Tivoli Enterprise Portal	2-33
Tivoli Enterprise Portal Take Action and system command authorization	2-34
Controlling Take Action capability	2-36
Summary	2-37

Table of contents

Unit 1: NetView for z/OS basics

Tivoli software

IBM

Unit 1: NetView for z/OS basics



© 2011 IBM Corp.

Introduction

In this unit, you learn about the key components and tools associated with NetView® for z/OS 6.1.

Objectives

Tivoli software

IBM

Objectives

When you complete this unit, you can perform the following tasks:

- Describe the packaging and installation of NetView
- Describe structure and components of NetView
- Use JCL procedures and parameters that are necessary to run NetView
- Use the NetView 3270 interface
- Customize NetView using CNMSTYLE
- Use the CNMSTYLE report generator
- Describe NetView operator administration
- Describe IP management with NetView

1-4

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Lesson 1: NetView overview

NetView for z/OS

Tivoli software



NetView for z/OS

- Helps maintain the highest degree of availability for IBM zEnterprise System networks
- Provides tools for managing and maintaining complex, multi-vendor, multi-platform networks and systems from a single point of control
- Has advanced correlation facilities to automate any network or system event
- Manages both TCP/IP and SNA networks
- Has management functions that work with other products
- Has open application programming interfaces
- Provides graphical displays and automation for managing systems and networks

1-6

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The NetView product has a long history. With each release, more functions have been added to better maintain networks. Earlier versions of NetView supported SNA networks. NetView now combines the support for both SNA and TCP/IP.

NetView program features

Tivoli software



NetView program features

- Increased network and system efficiency and availability
- Centralized management
 - TCP/IP and SNA network environments
 - Capability to help reduce the need for duplicate network management systems
- Enhanced operations and message management support
 - Capability for Improving and simplifying operator interactions
 - More control in automating and managing day-to-day operations
- Management as follows:
 - Larger networks
 - More components
 - More systems with fewer resources

1-7

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

IP management

Tivoli software

IBM

IP management

- SNA over IP
- Support for dynamic virtual IP addressing (DVIPA)
- Dynamic resource discovery
- IP connection management
- Packet trace collection and formatting
- Packet trace analysis
- Command support
- Automated responses to intrusions
- IPv6 support
- Support for Simple Network Management Protocol (SNMP) commands
- Network address translation (NAT)

1-8

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

NetView includes numerous commands for supporting IP. It also has been adapted for IPv6 and use of longer addresses.

Automation introduction

Tivoli software

IBM

Automation introduction

- You enable automated response to messages and events through the NetView Automation Table for reactive automation
- You can use user-written programs for automated responses
- Autotasks can receive messages and issue commands without the need for an operator
- You use message revision table (MRT) to revise message attributes, including text, or to suppress a message
- When necessary, you can create and trigger timer commands for proactive automation
- Event-correlation engine in UNIX System Services (USS) works with user-specified criteria
- You can use the Command Revision Table (CRT) in NetView to examine, modify, or reject MVS commands

1-9

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Automation capabilities of the NetView program facilitate and simplify operator interactions. Automation of responses to messages and events is enabled by the *NetView automation table*.

A correlation engine enables messages and **management services units (MSUs)** to correlate according to user-specified criteria. Message revision enables user-defined modification of attributes such as the following items:

- Color
- Route code
- Descriptor code
- Display
- Syslog settings
- Text of original z/OS messages (rather than copies)

Message revision actions include the following options:

- Revising messages before presenting to the system log, console, or automation
- Treating a message differently from others, depending on its source
- Suppressing messages entirely
- Automating only

The message revision table can override actions taken by the z/OS **message processing facility (MPF)** and can replace the MPF. It provides statistics and usage information and a test mode, and is active even when the NetView program is not. The message revision table is controlled by the NetView system programmer rather than by the z/OS system programmer.

User-written command lists and command processors can be used for accomplishing a complex operation by using a single command. These command lists can also perform complex procedures without operator intervention.

Timer commands initiate automated actions. Both operators and automation procedures can issue timer commands to schedule other commands, command lists, and command processors. Timer commands can occur at a specified time, after a specified delay, repeatedly after specified intervals, or in complex, timed combinations.

Autotasks are **operator station tasks (OSTs)** that do not require a workstation or operator. Autotasks can receive messages and issue commands. Autotasks are limited only that they cannot run full-screen applications.



Important: Autotasks for automation can be started during NetView initialization. The automation table, command lists, command processors, and timer commands can all issue commands under the autotasks. Autotasks can receive messages and present them to the automation table or to installation-exit routines. Installation exits are user-written routines that take control of processing at certain points to alter the typical course of NetView processing.

Sysplex monitoring

Tivoli software

IBM

Sysplex monitoring

- Aids in management of complex system configurations and interactions
- NetView agent automatically discovers the following items:
 - All z/OS images in a sysplex
 - All IP stacks on each z/OS image
- Configuration data and status display for systems that are in a sysplex

1-10

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Each NetView domain automatically collects information about the local system where it is running. The information pertains to the z/OS system and sysplex, TCP/IP stacks, and NetView itself.

In sysplex management, the NetView program uses z/OS Cross-Systems Coupling Facility (XCF) services to facilitate resource discovery, forward management data, and implement a controlling NetView program. The controlling NetView program, known as the master NetView program, has a complete view of the sysplex.

The master program is a logical point for interacting with the NetView user interfaces, the NetView management console, and the Tivoli Enterprise Portal. The master program is also a logical point for managing a Resource Object Data Manager (RODM) data cache from for the sysplex.XCF group. Signaling services enable the NetView programs in the sysplex to inform each other about status and exchange messages with each other.

XCF services enable the NetView programs to be notified when NetView programs enter or leave the sysplex. Information includes which NetView program is the master program, and about the relationship between the master program and other NetView programs. The use of XCF services makes the NetView program a high-availability sysplex application.

An XCF group is a set of related applications that registers with z/OS XCF services by using the same group name. At any given time, only one NetView program can be the master program in an XCF group. NetView XCF groups are named DSIPLEXnn, where nn is the value of the

XCF.GROUPNUM statement in the CNMSTYLE member. The DSIXCFMT task processes commands for joining and leaving XCF groups. It also processes information provided by XCF exit routines for handling messages and group status changes.

CNMSTYLE, located in one of the DSIPARM data sets allocated to NetView, is the primary initialization member for NetView.

Other NetView programs in the sysplex that are members of the same DSIPLXnn group can be defined as backups for the master NetView program, known as master-capable. If the master NetView program is unavailable, one of the master-capable NetView programs can take over the role of master, depending on the rank defined for each master-capable program. NetView programs that are neither master nor master-capable are known as basic NetView programs. Basic NetView programs forward status to the master NetView program but cannot assume the role of master.

Problem management

Tivoli software

IBM

Problem management

- You can use TECROUTE in NetView PIPE PPI to send an event to Tivoli Directory Integrator for use by Tivoli Service Request Manager or HP-Peregrine ServiceCenter
- NetView can improve information technology (IT) service management

1-11

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Use the TECROUTE option of the NetView PIPE PPI stage to send an event from the NetView address space to the Alert Adapter. The Alert Adapter, which converts that event to an EIF event, is in the NetView Event/Automation Service (E/AS). The EIF event is forwarded to Tivoli Directory Integrator for use by Tivoli Service Request Manager (TSRM).



Note: To avoid flooding TSRM, the NetView Automation Table should filter out unneeded events, so that only events that warrant a service request are forwarded to TSRM.

NetView Enterprise Management Agent (EMA)

Tivoli software



NetView Enterprise Management Agent (EMA)

- Provides enterprise integration with Tivoli Enterprise Portal, a graphical user interface
- Provides availability and performance data in a single interface
- NetView EMA provides OSA,DVIPA, sysplex, packet traces, the NetView log, SNA connection data, IP connection data, and more to the Tivoli Enterprise Portal
- Integrates with information from Tivoli OMEGAMON XE products

1-12

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Other user interfaces are NetView management console, web application, NetView web services gateway, and 3270. This unit describes each interface.

OMEGAMON® XE products integrate with the NetView EMA by using dynamic workspace links. Discussion of these products occurs later.

Security

Tivoli software

IBM

Security

NetView includes provisions to control the following situations:

- User access to the NetView program
 - User IDs and passwords with mixed-case support for SAF
 - Display of station access restrictions with SAF
 - Password lengths that can have as many as 100 characters
- Access to commands and data sets
 - System Authorization Facility (SAF) products
 - NetView command authorization table (CAT)
- Restricted access to NetView Management Console (NMC)
 - Views
 - Individual resources within views

1-13

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Restricting NetView Management Console (NMC) views and resources within views is particularly useful for service providers. The service providers can grant an operator access only to the resources of one customer but not another. Discussion about the NMC interface occurs in the next unit.

In addition to commands, NetView keywords and keyword values can be secured. CNMSCAT2, a member included with the product, demonstrates how to add operators to groups and control command authorities by these groups.

For environments with SAF, a sample job stream named CNMSAF2 is available. CNMSAF2 contains various Resource Access Control Facility (RACF®) statements to define operator user IDs, groups, and command permissions. It requires modification before running.

Discussion of CNMSCAT2 and CNMSAF2 occurs later.



Note: If the RACF mixed-case password function is enabled and passwords are defined in mixed case, NetView leaves them unchanged. Otherwise, NetView passwords are converted to uppercase. This processing applies to all password handling.

NetView 6.1 information center publications

Tivoli software



NetView 6.1 information center publications

Using

User's Guide: NetView
User's Guide: NetView Enterprise Management Agent
User's Guide: NetView Management Console
User's Guide: Automated Operations Network
IP Management

Installing

Installation: Getting Started
Installation: Configuring Additional Components
Installation: Configuring Graphical Components
Installation: Configuring the GDPS Active/Active Continuous Availability Solution
Installation: Configuring the NetView Enterprise Management Agent
Installation: Migration

Customizing

Application Programmer's Guide
Automation Guide
Customization Guide
Programming: Assembler
Programming: Pipes
Programming: PL/I and C
Programming: REXX and the NetView Command List Language
Resource Object Data Manager and GMFHS Programmer's Guide
SNA Topology Manager Implementation Guide
Tuning Guide

Troubleshooting

Troubleshooting Guide

Reference

Administration Reference
Command Reference Volume 1 (A-N)
Command Reference Volume 2 (O-Z)
Data Model Reference
Messages and Codes Volume 1 (AAU-DSI)
Messages and Codes Volume 2 (DUI-IHS)
Security Reference
Licensed Program Specifications
Program Directory for IBM Tivoli NetView for z/OS US English
Program Directory for IBM Tivoli NetView for z/OS Japanese
Program Directory for IBM Tivoli NetView for z/OS Enterprise Management Agent

IBM Tivoli NetView for z/OS V6R1 Online Library

LCD7-4913, contains the publications that are in the NetView for z/OS library. The publications are available in PDF, HTML, and BookManager® formats.

1-14

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The web address for the NetView publications is publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.itnetviewforzos.doc/ic-homepage.html

Lesson 2: IBM Tivoli NetView packaging and installation

Packaging and installation of NetView host code

Tivoli software

IBM

Packaging and installation of the NetView host code

- Product deliverables are shipped only via CBPDO, ServerPac, SystemPac®
- CBPDO and ServerPac are offered for Internet delivery in countries where ShopzSeries product ordering is available
- Minimum supported z/OS release levels have changed for the installation and execution of NetView 6.1
- The NetView Enterprise Management Agent requires the use of the Installation and Configuration Tool (ICAT), which is used by OMEGAMON products
- NetView is delivered in SMP/E installable format
- The installed product consists of a base Function Modification Identification (FMID) and a language FMID
- The z/OS supported levels are 1.10 or newer

1-16

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

NetView 6.1 contains a base component and *national language support (NLS)* feature:

- Base components and parts are installed using System Modification Program/Extended (SMP/E) with the HNV610B FMID. The base component is necessary for all installations.
- Using JNV610E FMID installs the English-language feature. Using JNV610J FMID installs the Japanese-language feature. The NLS component is based on ordering English or Japanese. NetView supports the installation of one NLS FMID.

NetView EMA code: z/OS

Tivoli software



NetView EMA code: z/OS

- NetView Enterprise Management Agent (EMA) component expands support and integration in the Tivoli Enterprise Portal
- The NetView components include NetView EMA code
- The NetView for z/OS Installation: The Getting Started manual has information on preparing MVS for the NetView EMA
- Refer to Installation: Configuring the Tivoli NetView for z/OS Enterprise Management Agent

1-17

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The data set members, starting with the *KNA* prefix make up the agent code on the host. Three NetView data sets contain KNA parts CNMLINK, CNMSAMP, and SCNMAGNT. The NetView EMA host code is included in NetView for z/OS 6.1 but has its own HNV610A FMID.

The *NetView for z/OS Program Directory* for the NetView Agent contains information for the prerequisite product, IBM Tivoli Monitoring.

The NetView EMA code should be installed in the data sets where the IBM Tivoli Monitoring is installed. If not, the NetView EMA code needs to be moved after SMP/E installation to the IBM Tivoli Monitoring data sets.

IBM Tivoli Management Services 6.2.2 is a free product with NetView for z/OS 6.1 that must be ordered by product number 5698-A79. It contains the z/OS components that are required for the NetView EMA. It includes FMIDs HKDS622, HKLV622, HKCI310, and HKT1710. Minimum service level for IBM Tivoli Monitoring is Fixpack 3 (FP3). IBM Tivoli Monitoring code for running Tivoli Enterprise Portal Server and Tivoli Enterprise Monitoring Server on Windows®, Linux®, and Linux® for System z is also delivered with 5698-A79.

Product number 5698-A79 consists of the following FMIDs:

- HKDS622 Tivoli Enterprise Monitoring Server on z/OS V622
- HKLV622 TMS: Engine V622
- HKCI310 Configuration Assistance Tool V310
- HKT1710 File Transfer Enabler for z/OS V1710

NetView EMA data files: Distributed

Tivoli software

IBM

NetView EMA data files: Distributed

- Minimum IBM Tivoli Monitoring 6.2.2
 - Prerequisite FixPack 3 (FP3) or newer
 - Separately orderable and priced product
 - As of June 2011, the current IBM Tivoli Monitoring version: 6.2.2 FixPack 4
- The NetView EMA data files compact disc (CD) includes files for the following items:
 - Tivoli Enterprise Portal
 - Tivoli Enterprise Portal Server
 - Distributed Tivoli Enterprise Monitoring Server support

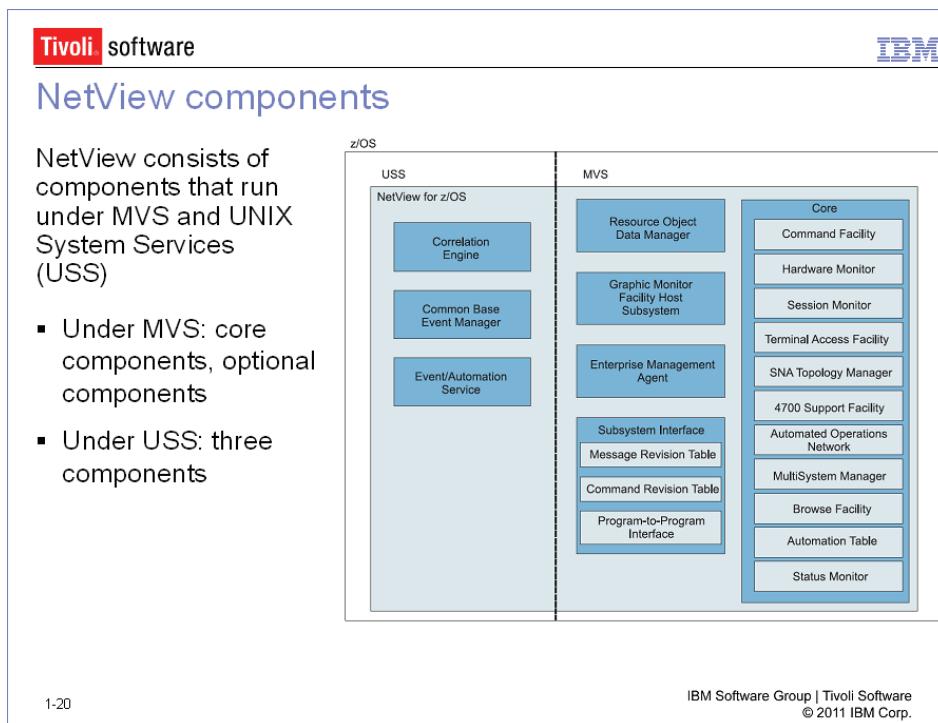
1-18

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The Tivoli Enterprise Monitoring Server database can be installed on either z/OS or distributed systems. DB2®, IBM Tivoli Monitoring 6.2.2 for Windows®, Linux®, and Linux® for System z files are included with NetView for z/OS 6.1. You can find IBM Tivoli Monitoring publications at publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.itm.doc_6.2.2fp2/welcome.htm.

Lesson 3: NetView structure and components

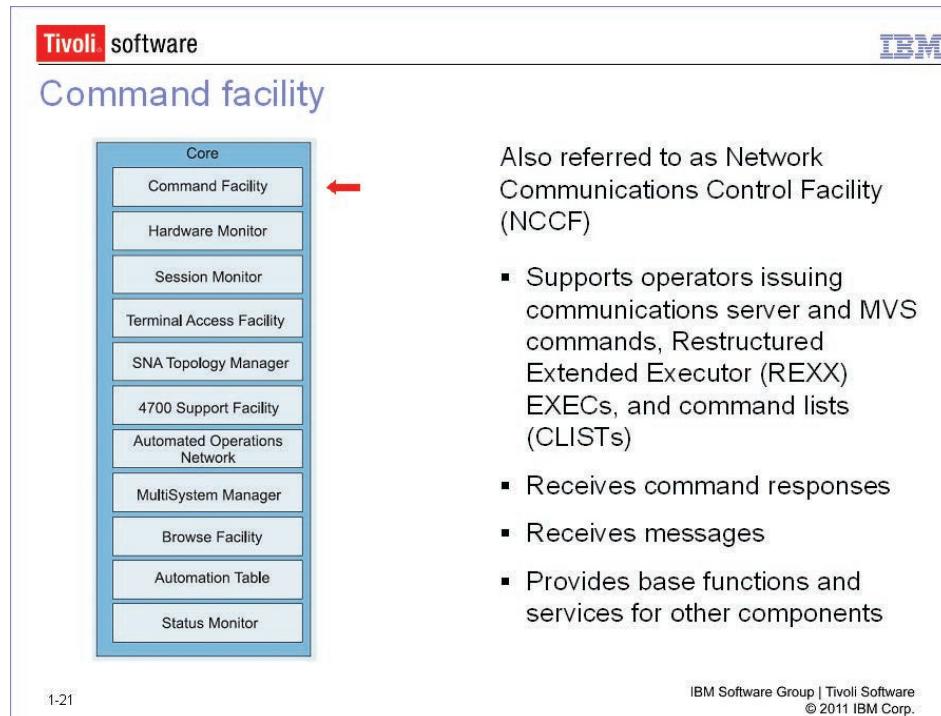
NetView components



The core components are in the column on the right side of the slide. Most of these components run under the NetView application address space. The subsystem interface (SSI) is a separate address space. The SSI is not necessary for running NetView, but might be necessary for some functions. These two address spaces together are executed for a functioning NetView system. All other components are optional.

In the NetView application, some of the core components are optional and do not need to be enabled.

Command facility



Network Communications Control Facility (NCCF) runs within the base NetView address space. Virtual Telecommunications Access Method (VTAM) is the Communications Server component that interacts with the Systems Network Architecture (SNA) communications. Communications Server also handles TCP/IP communications, which NetView uses. Base functions and services for other components include the following items:

- Intercomponent communications
- Communication with operating system
- Presentation services
- Database services
- Automation facilities

Hardware monitor

Tivoli software

IBM

Hardware monitor

Also referred to as Network Problem Determination Application (NPDA)

- Events and statistical data collected and displayed
- Probable causes
- Recommended actions
- Problem determination

Core
Command Facility
Hardware Monitor
Session Monitor
Terminal Access Facility
SNA Topology Manager
4700 Support Facility
Automated Operations Network
MultiSystem Manager
Browse Facility
Automation Table
Status Monitor

1-22

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Network Problem Determination Application (NPDA) runs within the base NetView address space. The hardware monitor creates a database made up of several record types: statistics, events, GMFALERTs, and alerts.

- Statistics are records of traffic and recoverable error counts that have been collected at certain resources and reported to the host system.
- GMFALERT records represent events that pertain to resources monitored by the NetView management console. If the NetView management console is not installed, the GMFALERT records are recorded in the hardware monitor database. The GMFALERT records are a subset of NetView management console event report records,
- Events are unexpected occurrences in network operation. An event can be created when the attempted activation of a resource fails. Resolution major vectors (X'0002'), which inform you that an alert was resolved, are also stored on the database as events.
- Alerts are events (including resolutions) that require attention.

Events and statistical data are also known as *alerts* and *MSUs*. You have several ways of sending this data to the hardware monitor:

- Forwarding an alert from one NetView program to another over an LUC session
- Sending a multiple domain support message unit (MDS-MU) over the MS transport to the ALERT-NETOP application
- Sending a control point management services unit (CP-MSU) or network management vector transport (NMVT) to the hardware monitor over the program-to-program interface

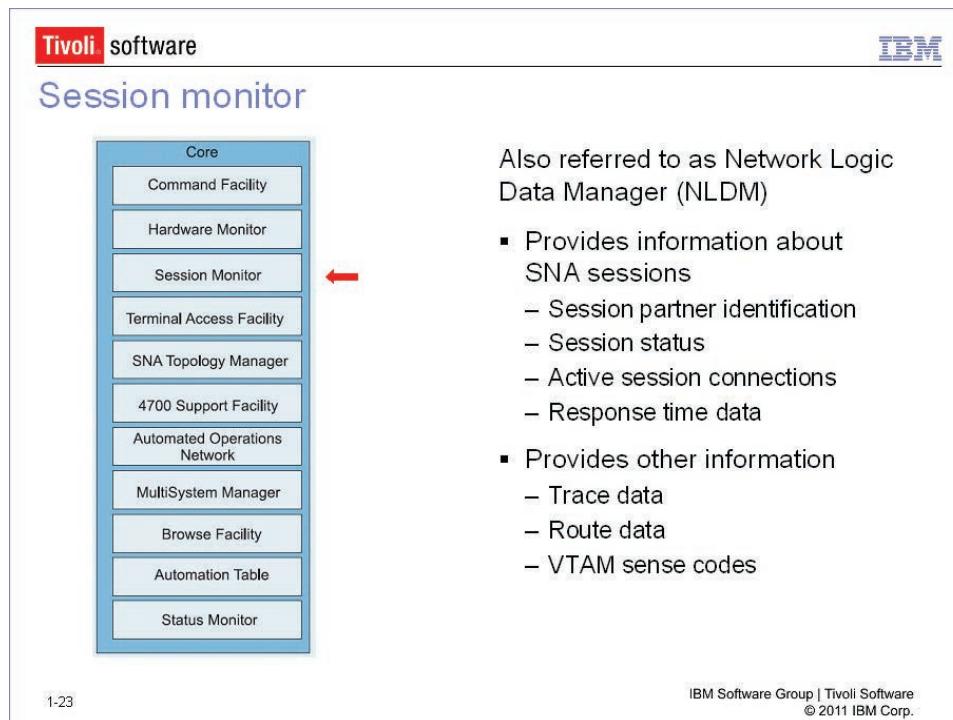
You can also use the hardware monitor for the following actions:

- Receiving a hardware-monitor problem record over the CNM interface. Example records are as follows:
 - NMVT
 - Record maintenance statistics (RECMS)
 - Record formatted maintenance statistics (RECFMS)
- Using the GENALERT command to generate a hardware-monitor record from within NetView.
- Receiving a system-format record for the hardware monitor (OBR, MDR, MCH, CWR, or SLH) from local MVS devices.

Many of the records that the hardware monitor receives go to the automation table during normal processing. The automation table can change filtering and highlighting attributes or issue automatic responses. Specifically, the records that go to the automation table are NMVTs, CP-MSUs, MDS-MUs, RECMSSs, and RECFMSs, collectively known as MSUs. The hardware monitor sends only MSUs that contain the following items:

- Alerts, key X'0000'
- Link events, key X'0001'
- Resolution, key X'0002'
- PD statistics, key X'0025'
- RECMSSs, encapsulated in a X'1044'
- RECFMSs, encapsulated in a X'1045'
- Link configuration data, key X'1332'

Session monitor



The Network Logic Data Manager (NLDM) component runs within the base NetView address space. This component includes information to help manage SNA sessions that run over IP when the customer uses Enterprise Extender (EE).

Terminal access facility

Tivoli software

IBM

Terminal access facility

- Numerous applications accessible from a single 3270 display station without logging off
- These applications locatable in different z/OS systems

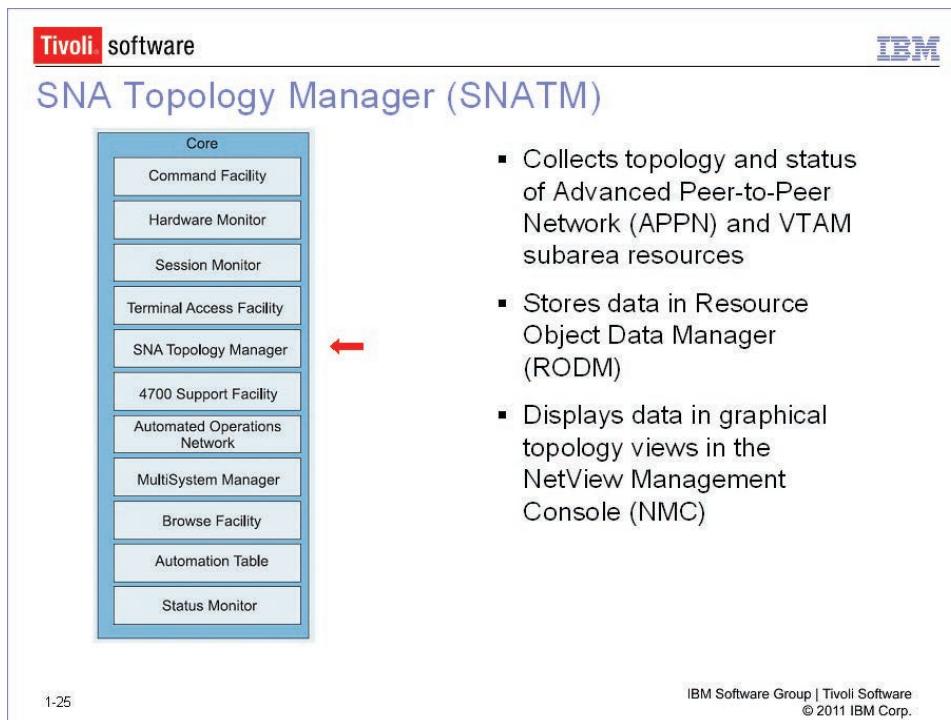
The diagram illustrates the structure of the NetView Core application stack. It consists of ten horizontal blue boxes stacked vertically. From top to bottom, the boxes are labeled: Core, Command Facility, Hardware Monitor, Session Monitor, Terminal Access Facility, SNA Topology Manager, 4700 Support Facility, Automated Operations Network, MultiSystem Manager, Browse Facility, Automation Table, and Status Monitor. A red arrow points to the 'Terminal Access Facility' box.

1.24

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

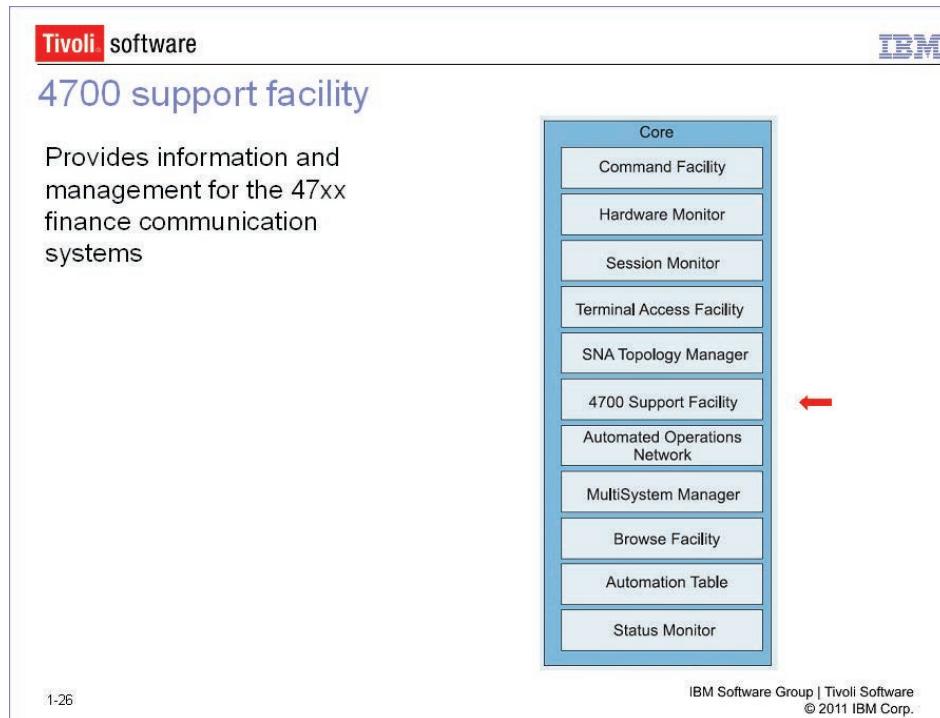
The **terminal access facility (TAF)** component runs within the base NetView address space. VTAM applications, such as TSO, CICS, and Information Management System (IMS) can be accessed from a single 3270 workstation. TAF supports both line-mode devices (LU1) and full-screen display devices (LU2).

SNA Topology Manager (SNATM)



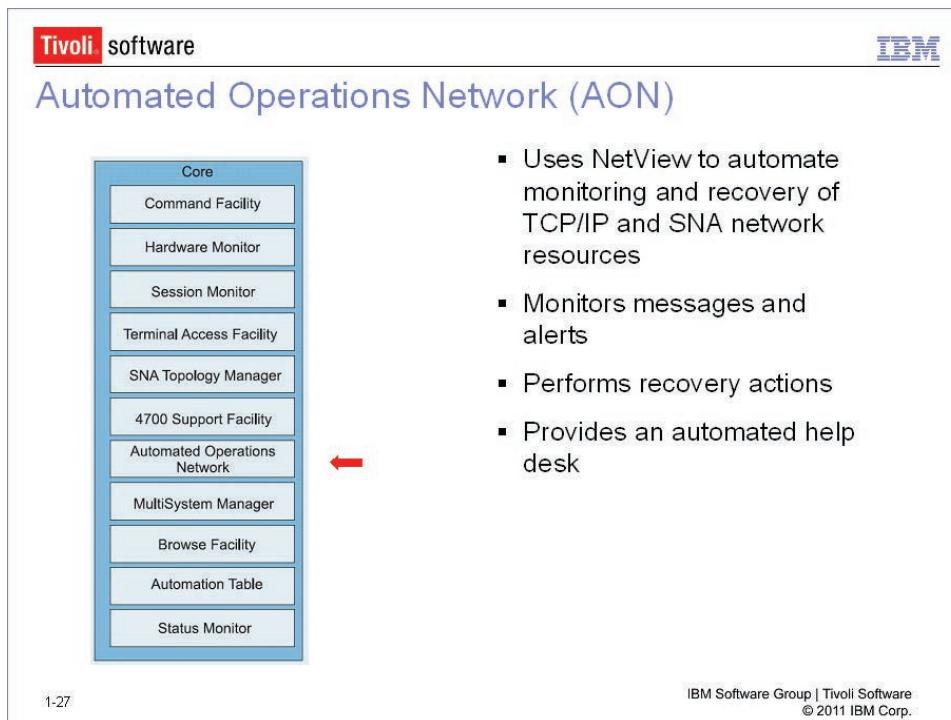
The SNA Topology Manager (SNATM) component uses a separate address space.

4700 support facility



The 4700 support facility component runs within the base NetView address space. The TARA panels are the means to communicate with the 4700 support facility. To view more information, access HELP TARA from the NetView console.

Automated Operations Network (AON)



The AON component runs within the base NetView address space.

MultiSystem Manager

Tivoli software

IBM

MultiSystem Manager

- Manages IP, other distributed resources
- Collects topology
- Collects status

The diagram illustrates the structure of the NetView Core. It is a vertical stack of ten rectangular boxes, each containing a component name. A red arrow points from the 'MultiSystem Manager' box back towards the left, indicating its relationship to the Tivoli software logo above. The components listed from top to bottom are: Command Facility, Hardware Monitor, Session Monitor, Terminal Access Facility, SNA Topology Manager, 4700 Support Facility, Automated Operations Network, MultiSystem Manager, Browse Facility, Automation Table, and Status Monitor.

1-28

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The MultiSystem Manager component runs within the base NetView address space. The MSM communicates with IBM Tivoli Network Manager via the MSM IBM Tivoli Network Manager agent. This component that is part of NetView for z/OS 6.1 runs in many different environments, such as the following operating systems:

- Windows®
- AIX®
- Linux
- Linux on zSeries
- Solaris®

Consult publications to find out the releases and other components that are needed.

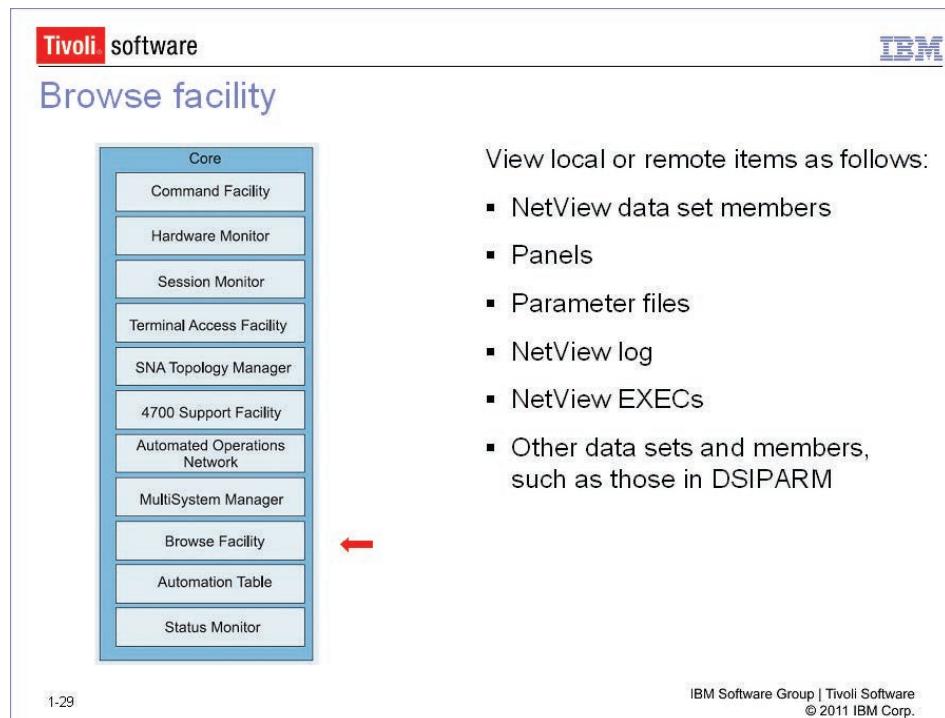
For the NetView for z/OS 6.1 component, an html file is on the workstation DVD LCD7-4914 with detailed information on system pre-requisites and installation instructions.

`file:///dvdroot/msm_nm_ip/msm_nm_ip_readme_en.html`

You can use MultiSystem Manager to manage the following networks:

- IP networks that IBM Tivoli Network Manager manages
- Any network that is supported by MultiSystem Manager Open topology agents. A toolkit is provided for third parties to write agents. The toolkit is available on the web.

Browse facility



The browse facility component runs within the base NetView address space.

Automation table

Tivoli software

IBM

Automation table

- Has processing options that you can specify for incoming messages or events
- Contains statements for defining actions to take, for example driving automation responses
- Is one of several components that is used in providing automation

```
graph TD; Core[Core] --> CF[Command Facility]; Core --> HM[Hardware Monitor]; Core --> SM[Session Monitor]; Core --> TAF[Terminal Access Facility]; Core --> STM[SNA Topology Manager]; Core --> 4700[4700 Support Facility]; Core --> AON[Automated Operations Network]; Core --> MSM[MultiSystem Manager]; Core --> BF[Browse Facility]; Core --> AT[Automation Table]; Core --> SM[Status Monitor];
```

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The automation table is a component that runs within the base NetView address space. This component is the heart of NetView automation services. It allows customers to select messages or events of interest, and take automated actions in response. Automated actions can be as simple as issuing a single command. Or it can be complex. One or more processes can be called to do an sophisticated analysis of a message or event. Additional possible action could be correlations of or with other messages or events.



Note: The automation table is discussed in the Automation Techniques module if included in this class.

Subsystem interface

Tivoli software

IBM

Subsystem interface

The diagram illustrates the structure of the Subsystem Interface. It consists of several stacked components:

- Resource Object Data Manager
- Graphic Monitor Facility Host Subsystem
- Enterprise Management Agent
- Subsystem Interface (highlighted with a red arrow)
- Message Revision Table
- Command Revision Table
- Program-to-Program Interface

▪ Used for receiving messages and entering system commands

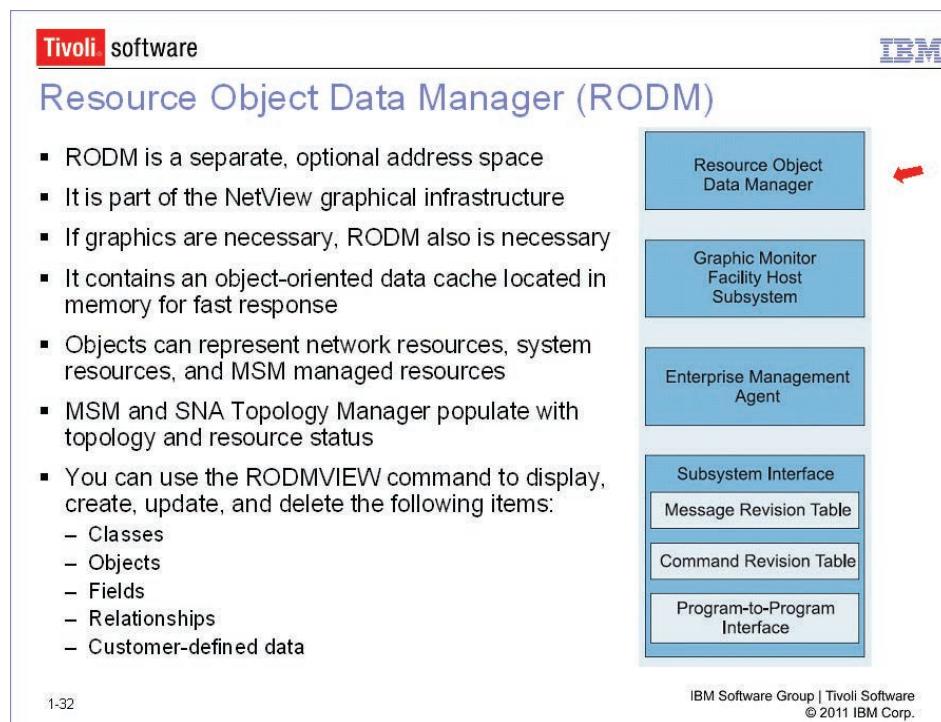
- With extended multiple console support (EMCS) consoles, the subsystem interface is used to receive commands, but not messages.
- Usable by NetView and other applications
- For each NetView system that uses the subsystem interface: NetView address space and subsystem address space
- With the message revision table (MRT)
 - z/OS messages can be intercepted before being displayed,
 - logged, automated, or routed through the network
 - Decisions can be made based on message ID, jobname, or other properties for taking an automated action or suppressing
- With the command revision table (CRT)
Intercept and revise z/OS commands and to make simple modifications inline, without needing to transfer the command to the NetView application address space
- With the program-to-program interface (PPI) address space
Application programs can communicate with NetView and other applications in the same host

1-31

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The subsystem interface component is a separate address space. Discussion about the MRT and CRT is in the *Automation* part of this class if offered. The command revision table (CRT) supersedes “MVS Command Management”. The NetView EMA uses the program-to-program interface (PPI) to communicate with the Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server.

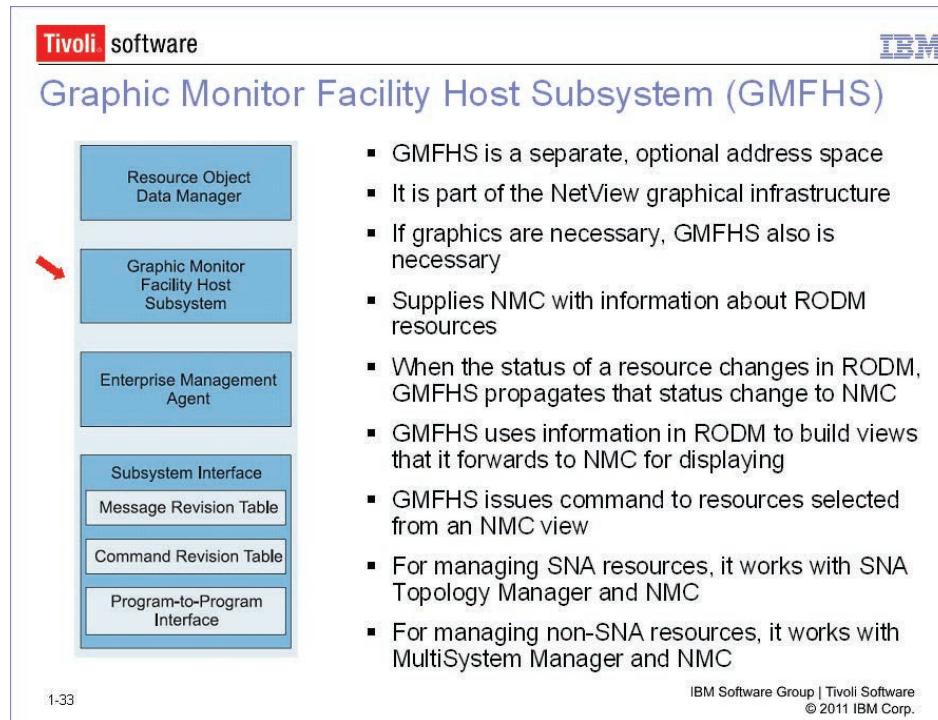
Resource Object Data Manager (RODM)



The Resource Object Data Manager (RODM) component is a separate address space. The RODMVIEW command is used with 3270 panels. The NMC displays data that is stored in the RODM, including topology and status.

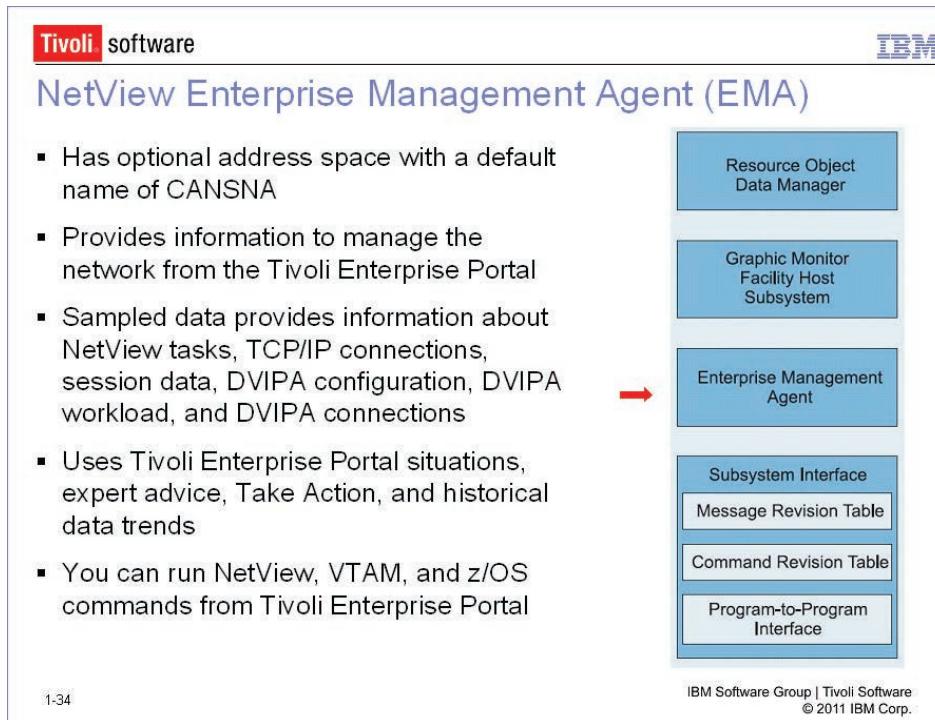
The **NetView Discovery Library Adapter (DLA)** extracts IP resource data from RODM, reformats it, and sends it to the IBM Configuration Management Database (CMDB). From CCMDB, it can be reconciled with data from other management products for a more complete picture of managed resources from multiple perspectives.

Graphic Monitor Facility Host Subsystem (GMFHS)



The Graphic Monitor Facility Host Subsystem (GMFHS) component is a separate address space. If you want graphics, you must use both RODM and GMFHS (along with NMC and MSM, or NMC, MSM, and SNATM).

NetView Enterprise Management Agent (EMA)



The NetView **Enterprise Management Agent (EMA)** component is a separate address space. This component provides data to one of the NetView user interfaces, the Tivoli Enterprise Portal.

Correlation engine

Tivoli software

IBM

Correlation engine

- Correlates multiple events over time
- You can correlate data, based on duplicate events, thresholds, the presence or absence of specific events, or user-specified information
- Extends automation capabilities
- Runs under UNIX Systems Services (USS)

```
graph TD; CE[Correlation Engine] --- CBE[Common Base Event Manager]; CBE --- EAS[Event/Automation Service]
```

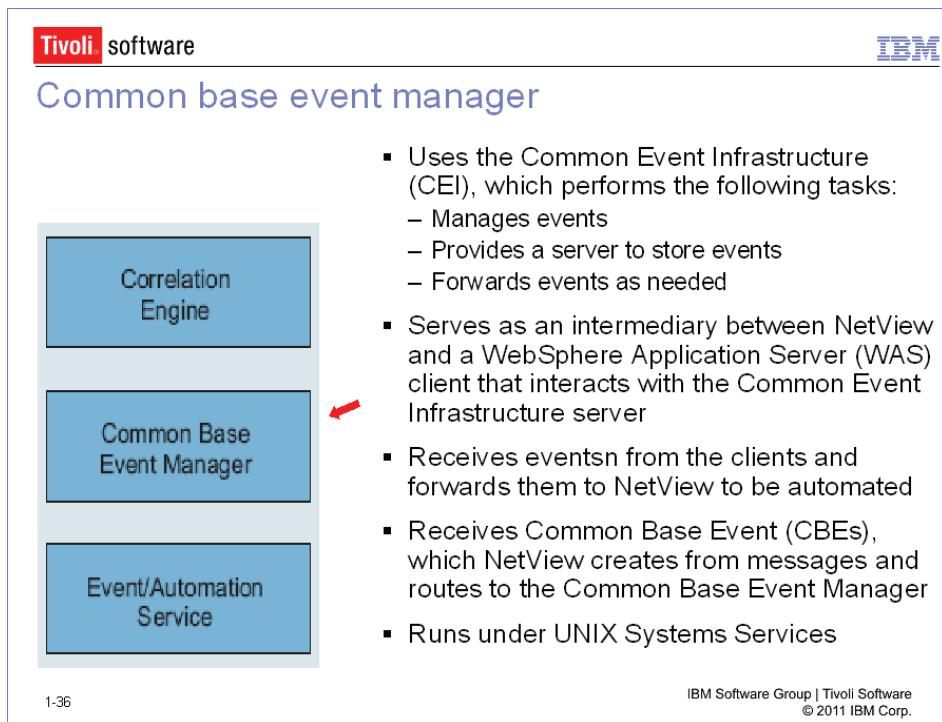
A vertical stack of three rectangular boxes. The top box is light blue and contains the text "Correlation Engine". A red arrow points from the text "Events are sent from the automation table to the correlation engine." to this box. The middle box is also light blue and contains "Common Base Event Manager". The bottom box is light blue and contains "Event/Automation Service".

1-35

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The correlation engine extends automation capabilities. Events are sent from the automation table to the correlation engine.

Common base event manager



NetView can both send and receive CBEs. When appropriate, the common base event manager sends data to the **WebSphere® Application Server (WAS)** client, which submits the event to the database. Connections for forwarding events are accepted from any number of clients.

Event/Automation Service (E/AS)

Tivoli software

IBM

Event/Automation Service (E/AS)

- Serves as a gateway for event data between the NetView for z/OS management environment, managers and agents that handle Event Integration Facility (EIF) events, and SNMP managers and agents.
- With this component, you can manage data from any of these management platforms
- Runs under UNIX System Services

```
graph TD; CE[Correlation Engine] --- CBE[Common Base Event Manager]; CBE --- EAS[Event/Automation Service];
```

1-37

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The Event/Automation Service (E/AS) component is a separate address space and is optional.

Lesson 4: NetView application procedure parameters and data sets

NetView procedure parameters

Tivoli software 

NetView procedure parameters

```
//NETVIEW EXEC PGM=&PROG,TIME=1440,  
//          REGION=&REG,  
//          PARM=(&BFSZ.K,&SLSZ,'&CNMDOMN',  
//                  '&DOMAINPW','&ARM','&SUBSYM','&NV2I','&TRSIZE'),  
//          DPRTY=(13,13)
```

Key parameters are as follows:

- PROG: The program is used to start NetView, typically BNJLINTX for network NetView or DSIMNT for System Automation for z/OS (SA z/OS)
- REG: The region size in K for the main task
- DOMAIN: NetView domain name
- NV2I: A unique two-character value used to build a symbolic for creating a unique parameter file name for a particular NetView. It substitutes the xx in CxxSTYLE. The default is NM

1-39 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

If you enable more components than those enabled by default, consider increasing the region size.



Note: If you specify a NV2I value of *xx*, for example, you must copy CNMSTYLE to C*xx*STYLE so that your editing can take place in C*xx*STYLE.

Other parameters are as follows:

- **DOMAINPW:** NetView domain password
- **BFSZ:** Buffer size in K
- **SLSZ:** Slot size, a fixed portion of a buffer to contain incoming error records
- **SUBSYM:** NetView symbolic substitution switch
 - **SUBSYM:** Enables symbolic substitution
 - ***NOSUBSYM:** Disables symbolic substitution and is the default
- **ARM:** Used for enabling (or not enabling) NetView for MVS automatic restart management
- **TRSIZE:** Size of NetView internal trace table.

Data definition names

Tivoli software



Data definition names

Some DDNAMES include the following terms:

- STEPLIB: The NetView load modules data set
- DSICLD: NetView REXX EXECs and command lists
- DSIPARM: NetView definition data set members and sense codes
- DSIPRF: NetView operator and autotask profiles

1-40

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Module loading occurs first from the STEPLIB data set or data set concatenation. Then the data sets that are defined in the z/OS PARMLIB member LNKLSTxx are searched. All NetView STEPLIB data sets must be authorized by the ***authorized program facility (APF)***.

Other DDNAMEs in the PROC are as follows:

- **SYSTCPD:** TCPIP parameters
- **CNMXLBIN:** TCP/IP translate data set TCPXLBIN
- **DSIOPEN:** NetView non-secured data sets
- **DSILIST:** Automation table listing
- **DSISARC:** Testing automation tables
- **DSIARPT:** Storage of output reports that are produced from running tests of the automation table
- **DSIVTAM:** VTAM library data sets (VTAMLST)
- **DSIMSG:** Message and translation members
- **BNJPNL1** and **BNJPNL2:** Hardware monitor panel data sets for central system and color maps
- **CNMPNL1:** Online help panels
- **DSILOGP** and **DSILOGS:** NetView Virtual Storage Access Method (VSAM) log data sets
- **DSITRCP** and **DSITRCS:** NetView VSAM trace log data sets
- **DSITCONP** and **DSITCONS:** TCP connection VSAM data sets
- **AAUVSPL** and **AAUVSSL:** Session monitor VSAM data sets
- **BNJLGPR** and **BNJLGSE:** Hardware monitor VSAM databases
- **BNJ36PR** and **BNJ36SE:** 4700 support facility VSAM data sets
- **FKPKTS:** Saved packet trace VSAM data set
- **CNMDVIPP** and **CNMDVIPS:** DVIPA workload statistics
- **DSIKPNL:** Central site control facility VSAM data sets
- **DSISVRT:** Save/Restore VSAM database
- **EZLSTAT:** AON automation status file
- **EZLPSWD:** AON password data sets for gateway session password management
- **EZLLOGP** and **EZLLOGS:** AON automation log data sets
- Several other print and utility data sets

NetView subsystem interface

Tivoli software 

NetView subsystem interface

```
//NETVIEW EXEC PGM=&PROG,TIME=1440,REGION=&REG.K,  
//           PARM=(&MBUF,&CBUF,'&DSIG','&MSGIFAC','&PPIOPT','&ARM',  
//           '&PFXREG',&P256BUF,&P4000BUF,&ROUTECDE),DPRTY=(13,13)
```

- STEPLIB is the NetView load module library data definition name (DDNAME)
- No other DDNAME statements are necessary

1-41 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

In NetView for z/OS 6.1, the following parameters are retired, meaning that they still need to be defined for migration purposes, but values are not used:

- MBUF
- CBUF
- DSIG
- PFXREG
- MSGIFAC

Other parameters are as follows:

- **ARM:** Used for enabling (or not enabling) NetView SSI automatic restart
- **PPIOPT:** Specification if you want to initialize the *program-to-program interface (PPI)* facility
- **P256BUF:** Specification of the number of 256-byte PPI buffers
- **P4000BUF:** Specification of the number of 4000-byte PPI buffers
- **REG:** Specification of the region size for the NetView subsystem buffer address space in kilobytes
- **ROUTECDDE:** Specification of the route code to be used for messages issued by the SSI address space

Lesson 5: NetView 3270 interface

NetView application program

Tivoli software 

NetView application program

- The 3270 sessions provide access to the core components and the command-line interface of the NetView program
- You can run commands to perform the following tasks:
 - Browse initialization parameters and data sets
 - Display TCP/IP information
 - Display VTAM status of devices
 - And so on
- You can run EXEC programs for automation, which is one of the most important uses of NetView
- To begin a NetView session, log on from a VTAM USS screen

1-43

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The 3270 interface is the first of the four user interfaces to NetView. Depending on your environment, a VTAM logon command can look like these examples:

```
logon applid(applid) logmode(logmode) data(data)
logon applid=applid logmode=logmode data='data'
```

Where:

- *Applid* is the name of the NetView application that you are logging on to.
- *LOGMODE* and *DATA* are optional parameters.
- *Logmode* specifies information about your workstation session.
- *Data* specifies information that is inserted in the OPERATOR ID and PASSWORD fields of the NetView logon panel.

Providing an operator ID and password

Tivoli software

IBM

Providing an operator ID and password

NN NN	VV	VV
NNNN NN EEEEEEE	TTTTTTTT VV	VV II EEEEEEE WW WW TM
NNNN NN EE	TT VV VV	II EE WW W WW
NN NN NNN EEEE	TT VV VV	II EEEE WW WWW WWW
NN NNNN EE	TT VV VV	II EE WWW WWW
NN NNN EEEEEEE	TT VVV	II EEEEEEE WW WW
NN NN	V	

5697-NV6 © Copyright IBM Corp. 1986, 2011 - All Rights Reserved
U.S. Government users restricted rights - Use, duplication, or disclosure
restricted by GSA ADP schedule contract with IBM corporation.
Licensed materials - Property of IBM Corporation

Domain = AOFDA

NetView V6RIMU

OPERATOR ID ==> or LOGOFF
PASSWORD ==>

PROFILE ==> Profile name, blank=default
HARDCOPY LOG ==> device name, or NO, default=NO
RUN INITIAL COMMAND ==> YES or NO, default=YES
Takeover session ==> YES, NO, or FORCE, default=NO

Leave password blank to change
Enter logon information or PF3/PF15 to logoff

- A NetView operator identification (ID) is necessary
- A password is typically required
- Other parameters are optional or can take defaults

1-44

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The NetView program queries the device for screen size and color attributes if the logmode specifies to issue the query. Otherwise, the NetView program uses the screen size that is specified in the logmode.

If the operator ID is already logged on and you want to take over the session, type “YES” as the Takeover session value. If you receive a message that the takeover is blocked, you can type “FORCE” as the Takeover session value.

NetView LISTA DSIPARM command

Tivoli software

IBM

NetView LISTA DSIPARM command

DDNAME	DATA SET NAME	MEMBER	DISP
DSIPARM	NV390.V681MO.WORKSHOP.DSIPARM		SHR,KEEP
	NV390.V681MO.USER.DSIPARM		SHR,KEEP
	NV390.DSIPARM		SHR,KEEP
	NV390.SAQPARM		SHR,KEEP

???

- Running the LISTA DSIPARM command results in display of the data sets allocated to the DSIPARM data definition (DD) statement in the NetView started task
- DSIPARM data sets are part of the initialization process of NetView
- DSIPARM data sets are partitioned data sets (PDS) and can be concatenated

1-45

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

NetView initialization begins with contents in the DSIPARM data sets. In the example, three data sets are concatenated to the DSIPARM DDNAME. The second and third data sets contain product-provided initialization members and samples. The first data set is used by this installation to contain modifications to DSIPARM members that are included with the NetView product.

DSIPARM data sets should not be allocated with secondary extents for disk space. If a data set expands into secondary space extents while NetView runs, problems can occur. Problems include incorrect locating and loading of members from the data set.

Canzlog



The screenshot shows the 'Specify Canzlog Filters' panel of the Tivoli Canzlog application. The 'Jobname:' field contains 'autossi'. Other fields like 'MSGID:', 'Jobid:', 'ASType:', etc., are empty. Below the filter settings, there are two search fields: 'Text - case sensitive; faster search:' and 'Text - case insensitive; slower search:'. A 'Name:' field has 'autossi' entered. A 'Remark:' field is also present. At the bottom, there's a note 'TO SEE YOUR KEY SETTINGS, ENTER 'DISPFK''. A 'CMD=> save' button is at the bottom right.

Tivoli software

Canzlog

Display

- MVS messages
- NetView messages
- Broadcast messages
- DOMs
- Command echoes
- Trace and audit messages

1.46

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Canzlog (consolidated audit, NetView, and z/OS log) shows all audit, system, and network messages in one place.

Slide shows canzlog filter panel. In this example we set a filter called autossi and save it. The filter looks for messages related to jobname AUTOSSI.

Canzlog ..

Tivoli software IBM

Canzlog ..
BR AUTOSSI

```
Canzlog : FILTER=AUTOSSI TAG=(NVMSC,MVSMSC) 07/13/11 13:31:23 -- 13:31:41
13:31:23 $HASP100 AUTOSSI ON STCINRDR
13:31:23 IEF695I START AUTOSSI WITH JOBNAMES AUTOSSI IS ASSIGNED TO USER NETVT
13:31:23 $HASP373 AUTOSSI STARTED
13:31:23 IEF403I AUTOSSI - STARTED - TIME=13.31.23
13:31:23 CNM226I NetView Program to Program Interface initialization is complet
13:31:23 CNM541I NetView subsystem AUTO is fully functional
13:31:41 IEF695I START AUTOSSI WITH JOBNAMES AUTOSSI IS ASSIGNED TO USER NETVT
13:31:41 IEF403I AUTOSSI - STARTED - TIME=13.31.41
13:31:41 CNM226I NetView Program to Program Interface initialization is complet
13:31:41 CNM541I NetView subsystem AUTO is fully functional
Bottom of Data
```

BR LOG will browse Canzlog based on filters set by DEFAULTS / OVERRIDE.
BR named_filter {optional_additional_filters} will browse Canzlog based on the specified parameters.
BR {filter_name} will browse Canzlog based on the parameters specified by the filter.
CANZLOG displays the filter panel. Subcommands are Replace, Save, Common, Task, and Delete

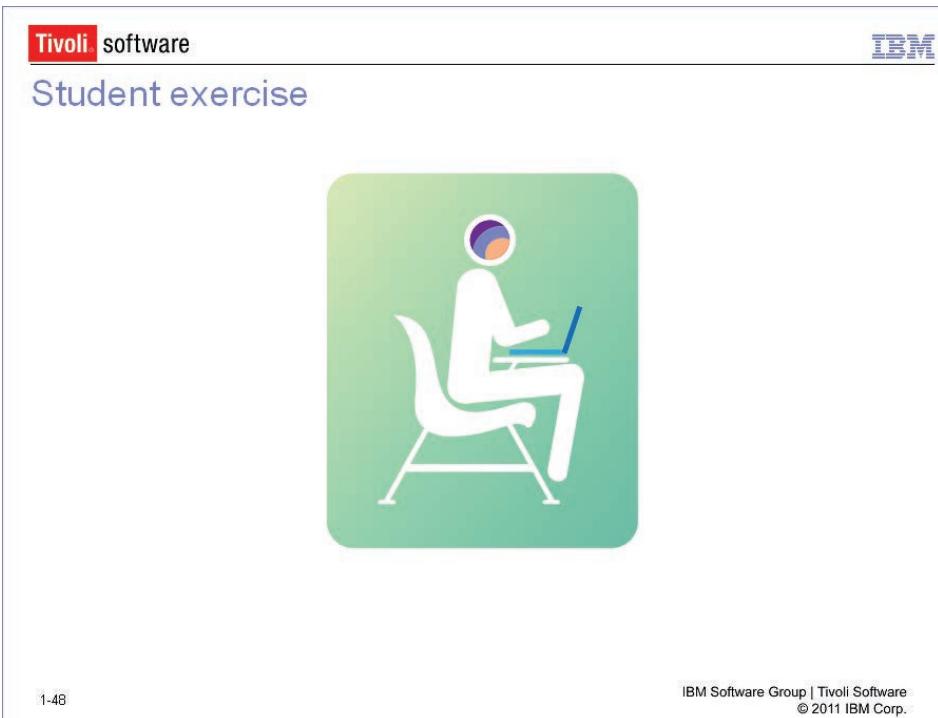
1-47 IBM Software Group | Tivoli Software
© 2010 IBM Corp.

Browsing a canzlog using filter AUTOSSI set in the previous slide.

For saving filters the following applies:

- Determine which NetView users will save task level filters. A dataset is required in DSIOPEN for each user, {CNMSTYLE.OPDSPREFIX}.{userid}.
- Users without this dataset will still be able to create and save task filters, but the filters will not be restored across logons. The default for CNMSTYLE.OPDSPREFIX is NETVIEW.OPDS
- All users will be able to create common filters as these are stored in the first dataset of DSIOPEN.

Student exercise



Open your *Student Exercises* book and perform Exercise 1-1.

Lesson 6: Customizing NetView by using CNMSTYLE

CNMSTYLE (1 of 2)

Tivoli software

IBM

CNMSTYLE (1 of 2)

- CNMSTYLE sets customization and configuration values during NetView initialization
- CNMSTYLE is a member of DSIPARM
- The sample CNMSTYLE member provided with the installation contains extensive descriptions and comments
- Using %INCLUDE, you can also include other DSIPARM members for CNMSTYLE initialization

1-50

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

CNMSTYLE is the first member that is read from the DSIPARM data set or from the concatenation of DSIPARM data sets. It is referred to as a *style sheet*.

The DSIPARM DDNAME can include many concatenated data sets. You can avoid member-loading problems by allocating DSIPARM data sets with only primary disk space and no secondary space.

CNMSTYLE (2 of 2)

Tivoli software

IBM

CNMSTYLE (2 of 2)

- The second and third characters of your stylesheet name can be variable. Use the &NV2I symbolic in the NetView application startup procedure to set the value for the following two-character positions:
 - The default value NM causes loading of CNMSTYLE
 - Setting &NV2I = LA causes loading of CLASTYLE
- If the C&NV2I.STYLE member cannot be found, NetView reads CNMSTYLE instead by default.
 - Do not make changes directly to the CNMSTYLE member
 - Make changes in CxxSTUSR or CxxSTGEN
- If statements are duplicated in CNMSTYLE and in any included members, the last of the duplicated statements applies

1-51

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The CNMSTYLE that is included with the NetView product can be used right after installation. Some parameters might require modification, but do not modify these in CNMSTYLE. Instead, copy the parameters to your CxxSTUSR member, which is to contain your changes. The data set that contains the CxxSTUSR member must be in a data set that is concatenated to the DSIPARM DDNAME. This data set must precede the data set that contains the CNMSTYLE that is supplied by NetView.

If &NV2I is specified incorrectly and C&NV2I.STYLE cannot be located in the DSIPARM PDS concatenation, CNMSTYLE is loaded instead. However, some unexpected results can occur. If %INCLUDE statements are defined in CNMSTYLE using &NV2I, such as %INCLUDE C&NV2I.STGEN, these members might not be located because the &NV2I value is incorrect.

Included members from DSIPARM (1 of 2)

Tivoli software



Included members from DSIPARM (1 of 2)

- CNMSTPWD
 - You can use this member to include vital product data (VPD), VSAM, and access-control-block (ACB) passwords
 - You can use the READSEC command to protect CNMSTPWD from being displayed by the BROWSE command
- CNMSTNXT (commented out in CNMSTYLE)
 - Includes modifiable CNMSTYLE statements by release
 - Comes with NetView for documentation purposes and as reference for someone migrating from a modified CNMSTYLE
- CNMSTASK (NetView-provided task statements, not to be modified)
 - If you want to change task statements, you can include the changes directly in CxxSTGEN
 - The task statements in CxxSTGEN override those provided in CNMSTASK, depending on the order of inclusion of these members in CNMSTYLE
- CNMSTUSR
- CxxSTGEN

1-52

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

If you have modified CNMSTYLE, you can open CNMSTNXT and copy just the statements that are new or changed in the last release or releases. You can add these changes to your older, modified CNMSTYLE. Customers that never change the NetView CNMSTYLE should not need to do this.

Some tasks are defined in the sample CNMSTYLE member with INIT=N (or INIT=NO) in CxxSTASK. If you plan to change these to INIT=Y (or INIT=YES), copy these to CxxSTGEN and change the statements in that member.

Discussion of CNMSTUSR and CxxSTGEN members occur later in this lesson.

Included members from DSIPARM (2 of 2)

Tivoli software



Included members from DSIPARM (2 of 2)

- CNMSTIDS
 - Includes statements for automating the detection of intrusions
 - You are to review this member if you are enabling Intrusion Detection Services (IDS) support
- CNMSTACT
 - Includes statements for the ACTIVEACTIVE subtower, LIFELINE or REPLICATION, specified.
- CNMSTTWR
 - Includes style statements from non-NetView towers
 - You are not to edit this member unless specifically instructed by documentation for a tower you are installing
- CNMSTWBM
 - Includes webmenu statements for controlling the web user interface

1-53

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

CNMSTTWR includes AOFSTYLE if TOWER=SA is specified in CNMSTYLE. SA is not a NetView tower.

Included member CNMSTUSR from DSIPARM

Tivoli software



Included member CNMSTUSR from DSIPARM

- You can add modifications to %INCLUDE member CxxSTUSR for changes common to the entire installation or enterprise
- You can include global (enterprise) definition statements that override statements in CNMSTYLE
- You can use %INCLUDE in this member to include other DSIPARM members
- You can make global (enterprise) changes to member CNMSTUSR, and then copy the modified CNMSTUSR to each NetView system
- You can use Data REXX logic

1-54

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

CNMSTUSR is a sample member that allows you to include any style statements that you write into CNMSTYLE. You can customize NetView and other product style statements. A %INCLUDE CNMSTUSR statement is in CNMSTYLE to include this member. You can copy this member to your own DSIPARM data set. You can change The second and third characters of the member name can be changed and addressed by a variable. This course refers to the two characters as *xx*. (The *xx* is the value of &NV2I, which is initially set to NM.)

Discussion of Data REXX logic, which is used to provide conditional logic in some style sheet members, occurs later.

Included member CxxSTGEN from DSIPARM

Tivoli software

IBM

Included member CxxSTGEN from DSIPARM

- CxxSTGEN is usable for system-specific changes, where xx is the value of &NV2I, initially set to NM
- You code all override statements for CNMSTYLE and CxxSTUSR in this member
- You can use Data REXX logic
- You can use %INCLUDE to include other DSIPARM members
- Duplicate statements found in CxxSTGEN override earlier statements in CNMSTYLE and CxxSTUSR
- Duplicate statements found in CxxSTUSR override earlier statements found in CNMSTYLE

1-55

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The CNMSTGEN member is provided by NetView with only comments. A %INCLUDE CNMSTGEN statement is in CNMSTYLE to include this member. Use this member for organizational purposes, such as adding system-specific modifications. The xx is the value of &NV2I, which is initially set to NM.

You can copy this member to your own DSIPARM data set. Use CxxSTGEN to tailor the style sheet when it is not convenient to have separate copies of CxxSTYLE. If duplicate statements are found during style sheet processing, the last occurrence of the statement is used.

Discussion of Data REXX logic, which is used to provide conditional logic in some style sheet members, occurs later.

CNMSTYLE changes during NetView execution

Tivoli software



CNMSTYLE changes during NetView execution

- The RESTYLE command can set and activate some changes without recycling the NetView program
 - Hardware monitor
 - Session monitor
 - Web interface
 - NetView Resource Manager
 - Visual BLDVIEWs
 - Various global variable updates
 - And many more
- With some exceptions, changes made to style sheet members while NetView is running become effective when recycling NetView

1-56

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The RESTYLE command reads the style sheet and sets new values pertaining to the specified keyword.

Data REXX logic

Tivoli software

IBM

Data REXX logic

- The logic is not supported in CNMSTYLE member
- You can use the logic in %INCLUDE members defined in CNMSTYLE
- You can add the logic to CxxSTUSR or CxxSTGEN member to conditionally process definition statements, based on whether a particular tower is enabled or not
 - %> IF tower('towername') THEN
 - %> DO;
 - definition statements
 - %> END;
- Data REXX files must begin with either a /* %DATA */ or /* %LOGIC */ statement
- The first column must begin with %> for the characters
- Lines beginning with a blank are considered continuation statements
- You can code the logic to conditionally change style sheet statements in CxxSTUSR or CxxSTGEN, such as enabling or disabling towers and subtowers

1-57

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The publication *IBM Tivoli NetView for z/OS Programming: REXX and the NetView Command List Language* provides more information about Data REXX.

Towers and subtowers (1 of 6)

Tivoli software

IBM

Towers and subtowers (1 of 6)

- Tower statements activate NetView components
- Tower statements are examined early in the NetView initialization process:

```
TOWER = *SA *AON *MSM *Graphics NPDA NLDM TCPIPCOLLECT
*AMI *TARA *DVIPA *TEMA *IPMGT *NVSQA DISCOVERY
*ACTIVEACTIVE
```
- Removing the asterisk enables that tower or function
- If multiple tower statements exist, the last one processes
- Subtower statements can enable specific components within a tower

```
TOWER.TCPIPCOLLECT = TCPCONN PKTS
```
- Modified tower statements are recognized only after NetView restarts

1-58

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

To update tower or subtower statements, copy the statements to CxxSTUSR or CxxSTGEN.

Towers and subtowers (2 of 6)

Tivoli software



Towers and subtowers (2 of 6)

- SA: Enables SA for z/OS
- AON – Enables network automation (AON component) and has these subtowers:
 - SNA: SNA automation (AON/SNA)
To also enable AON/SNA X.25 support, remove the asterisk (*) from the following statement:

*TOWER.AON.SNA = X25

- TCP: TCP/IP automation (AON/TCP)
To also enable Intrusion Detection Services (IDS) support, remove the asterisk (*) from the following statement:

*TOWER.AON.TCP = IDS

1-59

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

IBM Tivoli System Automation for z/OS (SA z/OS) is a separate product. Because it is not part of the NetView product, the subtowers that it uses are not listed.

Example of the tower statement for AON is as follows:

TOWER = AON

Examples of the AON subtower statements are as follows:

TOWER.AON.SNA = X25
TOWER.AON.TCP = IDS

Towers and subtowers (3 of 6)

Tivoli software



Towers and subtowers (3 of 6)

- MSM: Enables the MultiSystem Manager.
Subtowers:
 - ITNM: IBM Tivoli Network Manager component
 - OPN: Open component
- Graphics: Enables the NetView graphical infrastructure
Subtower: SNATM (SNA Topology Manager)
- NPDA: Enables the hardware monitor
- NLDM: Enables the session monitor

1-60

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The graphics tower enables NMC, RODM, and GMFHS.

Towers and subtowers (4 of 6)

Tivoli software

IBM

Towers and subtowers (4 of 6)

- TCPIPCOLLECT: Enables the collection of TCP/IP connection and packet trace data from IBM Communications Server for z/OS, with these subtowers
 - TCPCONN: Enables the collection of TCP/IP connection data
 - PKTS: Enables the collection of TCP/IP packet trace data
- AMI: Enables the Application Management Instrumentation
- TARA: Enables the 4700 support facility
- DVIPA: Enables the collection of dynamic virtual IP address (DVIPA) definition and status data. Subtowers:
 - DVTAD: Distributed DVIPAs
 - DVCONN: DVIPA Connections
 - DVROUT: VIPA Routes and DDVIPA Connection Routing

1-61

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

If TCPIPCOLLECT and DVIPA are enabled, this information can eventually be displayed through the Tivoli Enterprise Portal user interfaces to NetView.

More information about the ***application management instrumentation (AMI)*** are in members DSIAMIAT and DSIAMII.

DSIAMIAT is an automation table for AMI. The member that is provided with NetView has the capability of routing messages to another NetView. Such capabilities are as follows:

- It can route messages to a message adapter.
- Messages may be sent to Tivoli Enterprise Console by a message adapter.
- Messages can be sent to a NetView management console topology server.

DSIAMII is used for customizing monitor default threshold specifications and polling intervals as appropriate for your environment.

Towers and subtowers (5 of 6)

Tivoli software



Towers and subtowers (5 of 6)

Tivoli Enterprise Monitoring Agent: Enables the NetView Enterprise Management Agent and uses the following subtowers:

- HEALTH: Enables the collection of NetView health data
- CONNACT: Enables the collection of data on active TCP/IP connections
- CONINACT: Enables the collection of data on inactive TCP/IP connections
- SESSACT: Enables the collection of data on active sessions
- DVDEF: Enables the display of DVIPA definition and status data
- DVTAD: Enables the collection of DVIPA sysplex distributors and distributor targets data
- DVCONN: Enables the collection of data on DVIPA connections
- SYSPLEX :Enables the collection of stack configuration and status data
- TELNET: Enables the display of discovered TELNET servers and ports
- DVROUT: Enables the display of discovered VIPA routes and DDVIPA connection routing
- OSA: Enables the display of OSA channels and ports
- HIPERSOCKETS: Enables the display of discovered HiperSockets
- ACTIVEACTIVE: Shows availability and performance metrics for workload distribution and replication capture and apply servers

1-62

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The Tivoli Enterprise Monitoring Agent tower enables communication not only between core NetView and the NetView EMA, but also the entire NetView EMA function. NetView communicates with the Tivoli Enterprise Portal through the NetView EMA. The NACMD command is used for communicating between the NetView and NetView EMA address spaces. If the Tivoli Enterprise Monitoring Agent tower is not enabled, the NACMD command fails.



Note: The SESSACT subtower is supported in only one NetView program per z/OS logical partition (LPAR).

Towers and subtowers (6 of 6)

Tivoli software



Towers and subtowers (6 of 6)

- IPMGMT: Perform active monitoring without the need for the following AON subtowers:
 - ACTMON
 - IDS
- NVSOA: NetView Web Services Gateway
- DISCOVERY: Discovers telnet servers and interfaces. Subtowers:
 - TELNET
 - INTERFACES
 - OSA
 - HIPERSOCKETS
- ACTIVEACTIVE: GDPS Active/Active. Subtowers:
 - LIFELINE
 - REPLICATION

1-63

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The IPMGT ACTMON subtower can be used for performing active monitoring or IP resources without the need for the AON Tower. The IPMGT IDS subtower can be used for Intrusion Detection automation without the need for the AON tower. These towers are mutually exclusive with the AON TCP tower.

Tower and subtowers for Tivoli Enterprise Monitoring Agent

Tivoli software

IBM

Tower and subtowers for Tivoli Enterprise Monitoring Agent

- You do not enable the Tivoli Enterprise Monitoring Agent tower unless you are installing the NetView EMA
- You need only one NetView EMA per logical partition (LPAR)
- Installing of NetView EMA requires new address spaces to run CANSDSST and CANSNA
 - CANSDSST is the default procedure name for the Tivoli Enterprise Monitoring Server, if the Tivoli Enterprise Monitoring Server is running on a z/OS system
 - CANSNA is the default procedure name for the EMA, which must run on a z/OS system

1-64

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The Tivoli Enterprise Monitoring Server does not need to run on a z/OS system, and it can run on a distributed system. Many z/OS installations run the Tivoli Enterprise Monitoring Server on z/OS systems because mainframe operating systems offer reliability, availability, and backup capabilities. In most cases, a remote Tivoli Enterprise Monitoring Server is needed on each z/OS image, as some agents can run only within a Tivoli Enterprise Monitoring Server. Example of this is OMEGAMON XE on z/OS agent.

Commands automatically run after initialization

Tivoli software

IBM

Commands automatically run after initialization

- You use the auxInitCmd statement in CxxSTUSR or CxxSTGEN
- You specify any number of commands or command lists to run
- The EBCDIC value that follows the auxInitCmd keyword determines the order that the commands run
 - auxInitCmd.A = MSG SYSOP,Auxiliary commands the beginning
 - auxInitCmd.AC = RESTORE TIMER
- AuxInitCmd commands run before any commands at any autotask

1-65

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

These auxInitCmd commands provide the initial step for triggering automation as NetView starts up. All commands for autotasks are queued, including both task initial command lists and commands that EXCMD sends. These commands run only after all auxInitCmds have completed. Messages are also queued. They do not route to automation nor logging until all auxInitCmd commands finish.

Setting up security

Tivoli software



Setting up security

You use SECOPTS statements to specify the following items:

- Operator security
- Command authority
- Span of control authority
- Web browser access

1-66

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The *IBM Tivoli NetView for z/OS Administration Reference* manual and the *IBM Tivoli NetView for z/OS Security Reference* manual explain the use of SECOPTS statements. The next slide briefly describes operands.

SECOPTS statements and options in CNMSTYLE

Tivoli software

IBM

SECOPTS statements and options in CNMSTYLE

SECOPTS.OPERSEC = NETVPW	SECOPTS.OPSPAN = NETV
*SECOPTS.OPERSEC = SAFCHECK	*SECOPTS.OPSPAN = SAF
*SECOPTS.OPERSEC = SAFPW	SECOPTS.SPANAUTH = *NONE*
*SECOPTS.OPERSEC = SAFDEF	*SECOPTS.SPANAUTH = TABLE.CNMSPAN2
*SECOPTS.OPERSEC = MINIMAL	*SECOPTS.SPANAUTH = VTAMLST.CNMSPAN1
SECOPTS.SURROGAT = NO	SECOPTS.SPANCHK = SOURCEID
SECOPTS.CMDAUTH = TABLE.CNMSCAT2	*SECOPTS.SPANCHK = TARGETID
*SECOPTS.CMDAUTH = SAF.CNMSBAK1	SECOPTS.WEBAUTH = PASS
*SECOPTS.CMDAUTH = SAF.PASS	*SECOPTS.WEBAUTH = CHECK
*SECOPTS.CMDAUTH = SAF.FAIL	SECOPTS.RMTAUTH = SENDER
*SECOPTS.CMDAUTH = SCOPE.CNMSCOP1	*SECOPTS.RMTAUTH = ORIGIN
SECOPTS.AUTHCHK = SOURCEID	
*SECOPTS.AUTHCHK = TARGETID	

1-67

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can use operands to the SECOPTS statements as follows:

- OPERSEC defines the method used for logging on to the NetView program.
- SURROGAT specifies if the NetView operator ID is checked to determine if it is a surrogate of a TSO user ID.
- CMDAUTH defines the method that NetView uses for protecting command usage.
- AUTHCHK specifies the user ID that is to be used when verifying command authorization.
- OPSPAN defines the method for determining the authority of an operator to start spans of control.
- SPANAUTH specifies the location of the span definitions for resources and views.
- SPANCHK specifies the operator ID that is used for defining span checking.
- WEBAUTH specifies if authorization checking is to be performed for operator access to the NetView Web server.



Note: RMTAUTH specifies the method to determine the operator ID that is used as the remote operator for security checks that are performed on RMTCMD and ENDTASK requests. RMTAUTH is used for only incoming requests, not for command security checking.

Student exercise

Tivoli software

IBM

Student exercise



1-68

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 1-2.

Lesson 7: CNMSTYLE report generator

CNMSTYLE report generator features

Tivoli software

IBM

CNMSTYLE report generator features

- You can analyze CNMSTYLE and its included members
- You can use the report to perform the following tasks:
 - List the %INCLUDE structure
 - Analyze multiple occurrences of statements within CNMSTYLE and its included members
 - Determine the value that is used during NetView initialization. (For statements that are listed multiple times in the report, the last statement listed is the one that is used for initialization.)
 - List the CNMSTYLE towers that are enabled
 - Analyze initialization statements for a particular function
- You can run the report by using the sample CNMSJCRG batch job that is in the INSTALL data set
- By default, the report is written to the data set that is allocated to the DSIWRIT DDNAME

1-70

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The CNMSJCRG job runs outside of the NetView address space and runs the CNMECRG REXX program from the TSO terminal monitor program in batch mode. A sample job is located in &HLQ.INSTALL(CNMSJCRG).

CNMSJCRG requires the following data definitions and data sets:

- STEPLIB: The NetView CNMLINK data set from the current release, NETVIEW.V6R1M0.CNMLINK, by default
- SYSEXEC: The concatenated data set list of the NetView CNMCLST data sets from the current release
- DSIPARM: The concatenated data set list containing current release versions of CNMSTYLE. Ensure that the data set concatenation order matches the order that the NetView CNMPROC start procedure specifies.
- CNMPNL1: The NetView CNMPNL1 data set from the current release, NETVIEW.V6R1M0.CNMPNL1.
- DSIWRIT: The destination for the report. By default, this is written to NETVIEW.V6R1USER.CNM01.DSILIST(CNMCRG). If the member already exists, CNMCRG is copied to CNMCRGBK before overwriting CNMCRG.

CNMSJCRG job

Tivoli software

IBM

CNMSJCRG job

- The following example shows keyword parameters for the REXX EXEC CNMECRG that is coded in the batch job CNMSJCRG

```
CNMECRG
        TASKS=NO
        &NV2I=NM
        &DOMAIN=CNM01
        &CNMTCPN=TCPIP
        &CNMRODM=RODMNAME
        &CMNETID=NETA
        &MYSYMBL=' A B C '
```

- The CNMSTYLE report includes the following sections:
 - General information and CNMSTYLE statements that pertain to all of NetView
 - CNMSTYLE statements that pertain to specific functions of NetView
 - auxInitCmd statements and user-defined statements
 - Data REXX statements within CNMSTYLE
- Because the TASKS parameter is set to NO, the report in this example does not include CNMSTASK statements

1.71

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The CNMSTYLE report is written to a member of a PDS. The length of each line in the report is 80 bytes.

Typical CNMSJCRG job

The screenshot shows a terminal window titled "Typical CNMSJCRG job". The window has a menu bar with "File", "Edit", "Edit_Settings", "Menu", "Utilities", "Compilers", "Test", and "Help". Below the menu is a command line area with the following text:

```
VIEW    USER.INIT(CNMSJCRG) - 01.02          Columns 00001 00072
***** **** Top of Data ****
000001 //CNMSJCRG JOB 'ACCOUNTING INFORMATION', 'STYLE REPORT GENER',
000002 // CLASS=A, MSGCLASS=A, MSGLEVEL=(1,1), NOTIFY=IBMUSER
000003 //STEP1 EXEC STYLERPT
000004 /*
000005 // CALL THE CNMECRG REXX EXEC AND PASS THE INPUT PARAMETERS
000006 /*
000007 //IKJEFT01.SYSTSIN DD DATA
000008 PROF MSGID
000009 CNMECRG
000010  TASKS=YES
000011  &DOMAIN=AOFDA
000012  &NV2I=N
000013  &CNMTCPN=TCPPIP
000014 /*
000015 //
***** **** Bottom of Data ****

Command ==> [ ] Scroll ==> PAGE
F1=Help F3=Exit F5=Rfind F6=Rchange F12=Cancel
```

At the bottom of the window, there is a status bar with the following text: "Command ==> [] Scroll ==> PAGE" and function key definitions: F1=Help, F3=Exit, F5=Rfind, F6=Rchange, F12=Cancel.

The text "The report that results from running this job is on the slides that follow" is displayed below the terminal window.

The report that results from running this job is on the slides that follow

1-72

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

CNMSJCRG job keyword parameter notes are as follows:

- CNMECRG is the invoked command.
- Keyword parameters and values follow CNMECRG.
- Each keyword must be on a separate line.
- A value must not continue into a second line.
- All characters that are typed on a line are interpreted as input to CNMECRG.
- If a keyword parameter is specified more than once, the first value is used and all subsequent values specified are ignored.
- Input ends when either a blank line or a /* occurs.

CNMSJCRG job keywords are as follows:

- **TASKS=YES | NO.**
 - Specifies whether to include CNMSTASK statements in the report or not.
 - YES includes statements from CNMSTYLE %INCLUDE member CNMSTASK. YES is the default value.
 - NO excludes CNMSTASK statements.
- **&NV2I=xx.**
 - The default value for *xx* is NM.
 - If an incorrect value is specified, an error message is issued and the default value NM is used in the report.
 - If you use alphabetic characters, the characters convert to uppercase.
- **&symbolic_name= value.**
 - Provides the value of a system or NetView symbolic variable (&symbolic_name) that you are using in CNMSTYLE or its included members.
 - A symbolic parameter must be passed to CNMECRG to be resolved in the report.

The following return codes are set by CNMECRG:

- 0 Successful job completion, a file created in DSIWRIT.
- 4 Minor errors encountered during job, a file created in DSIWRIT.
- 8 Major error encountered during job, a file not created in DSIWRIT.



Note: For non-zero return codes, error messages are in the CNMSJCRG job log.

Details of the job output are in the next set of slides.

CNMSTYLE report: First section

Tivoli software 

CNMSTYLE report: First section

The first section contains general information

- The date and time of report creation
- The &NV2I symbolic variable value that is being used
- A nested listing of the members that CNMSTYLE includes
- A list of the CNMSTYLE towers that are enabled when NetView initializes
- A list of CNMSTYLE statements that apply to base NetView. An example is on the next two slides

1-73

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The format of the CNMSTYLE statements presented in the generated report includes the following fields:

- **Member:** Member name containing the statement
- **Line#:** Line number within the member where the CNMSTYLE statement is located
- **Indicators:** Information about the statement, formatted with R (Resolve indicator) and CCCCCC (Condition indicator)
- **Resolve:** Indication if the CNMSTYLE statement was modified by the report generator. A specification of Y indicates that the statement was modified.
- **Condition:** Indication that a condition is required for the listed CNMSTYLE statement to be active, such as a tower that must be enabled.
 - If only one tower is required to be enabled, the condition field is set to the required tower name.
 - If more than one tower must be enabled, or if another condition must be met, the condition field is set to four asterisks (****).
- **Statement:** Listing of the CNMSTYLE statement and its value.
 - Statements are modifiable by having values substituted into either the CNMSTYLE keyword or its value.
 - Values that contain passwords list in the report as four asterisks.

%INCLUDE structure

Tivoli software

IBM

%INCLUDE structure

File Edit Edit_Settings Menu Utilities Compilers Test Help

VIEW NV390.V6R1M0.USER.DSILIST(CNMCRG) - 01.00 Columns 00001 00072
Command ==> [] Scroll ==> CSR

***** Top of Data *****

000001
000002
000003
000004 DATE: 23 Jun 2011
000005 TIME: 08:54:28
000006
000007 &NV2I value: NM
000008
000009 %INCLUDE structure of: CNMSTYLE
000010
000011 CNMSTYLE
000012 CNMSTPWD
000013 CNMSTASK
000014 CNMSTIDS
000015 CNMSTACT
000016 CNMSTLIF
000017 CNMSTREP
000018 CNMSTTWR
000019 CNMSTWBM
000020 CNMSTUSR
000021 CNMSTGEN
000022
000023 Enabled Towers: NPDA NLDM TCPIPCOLLECT IPMGT
000024
000025 Statements for function: NetView General

1.74 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can view the output of the job with Interactive System Productivity Facility (ISPF) BROWSE. This is part of the first section of the report.

The display shows the order of members that are being included as follows.

- CNMSTYLE precedes an indentation for the next seven members, indicating that CNMSTYLE includes these seven members.
- CNMSTGEN precedes an indentation and three more members, indicating that CNMSTGEN includes these three members.

General NetView statements

Tivoli software 

General NetView statements

File	Edit	Edit_Settings	Menu	Utilities	Compilers	Test	Help
VIEW	NV390.V6R1M0.USER.DSILIST(CNMCRG)	- 01.00		Columns 00001 00072			
Command	==>			Scroll ==> CSR			
000025	Statements for function: NetView General						
000026							
000027	Member	Line#	Indicators	Statement			
000028							
000029							
000030	CNMSTYLE	217 Y		DOMAIN = CNM01			
000031	CNMSTGEN	7 Y		DOMAIN = AOFDA			
000032							
000033	CNMSTYLE	304		NetID = &CNMNETID.			
000034							
000035	CNMSTYLE	751		TOWER = *SA *AON *MSM *Graphics NPDA NLDM			
000036				TCPIPCOLLECT *AMI *TARA *DVIPA *TEMA *IP			
000037				*NVSQA DISCOVERY *ACTIVEACTIVE			
000038	CNMSTGEN	112 ****		TOWER = *SA *AON *MSM *Graphics NPDA NLDM			
000039				TCPIPCOLLECT *AMI *TARA *DVIPA TEMA *IPM			
000040				NVSOA DISCOVERY *ACTIVEACTIVE			
000041	CNMSTGEN	140 ****		TOWER = *SA *AON *MSM *Graphics NPDA NLDM			
000042				TCPIPCOLLECT *AMI *TARA *DVIPA *TEMA IPM			
000043				*NVSQA *DISCOVERY *ACTIVEACTIVE			
000044							
000045	CNMSTYLE	1630		CNMI = Yes			
000046	CNMSTGEN	156		CNMI = Yes			
000047							
000048	CNMSTYLE	585		SECOPTS.OPERSEC = NETVPW			
000049	CNMSTGEN	256		SECOPCS.OPERSEC = SAFPW			
000050							

1-75 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide shows the first section scrolled forward. The screen displays examples of original statements and their statements. The CNMSTGEN member has redefined values for DOMAIN, NETID, and CNMI in this extract of the report. If you specified DOMAIN in the NetView started procedure, the value in CNMSTYLE is ignored.

In this example, note the differences between the two TOWER statements. The second tower enables AON and DVIPA. It disables NLDM. With NetView style sheets, the last one applies. In other words, the last definition of the same statement or parameter in a statement is the one that is used.

CNMSTYLE report: Second section

Tivoli software

IBM

CNMSTYLE report: Second section

The second section of the report lists CNMSTYLE statements for specific NetView functions, such as the hardware monitor (NPDA) component

- If a CNMSTYLE statement applies to multiple NetView functions, that statement is listed for each applicable NetView function. For example, the TOWER statement applies to the hardware monitor, session monitor, and various other NetView functions
- Within a function, the most critical statements are listed first, followed by less critical statements
- NetView functions in the report are in alphabetical order

1.76

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Example of TCP/IP tower functions

Tivoli software IBM

Example of TCP/IP tower functions

```
File Edit Edit_Settings Menu Utilities Compilers Test Help
VIEW NV390.V6R1M0.USER.DSILIST(CNMCRG) - 01.00 Invalid character string
Command ==> █ Scroll ==> CSR
001959
001960 Statements for function: TCP/IP Alert Receiver
001961
001962 Member Line# Indicators Statement
001963 -----
001964 CNMSTASK 147 TASK.DSIRTTR.MOD = DSIZDST
001965 CNMSTASK 148 TASK.DSIRTTR.MEM = DSIRTTTD
001966 CNMSTASK 149 TASK.DSIRTTR.PRI = 6
001967
001968 CNMSTYLE 1734 TASK.DSIRTTR.INIT = N
001969 CNMSTGEN 167 **** TASK.DSIRTTR.INIT = N
001970 CNMSTGEN 209 **** TASK.DSIRTTR.INIT = N
001971
001972 CNMSTYLE 2221 Y RTT.TCPANAME = TCPIP
001973 CNMSTYLE 2222 RTT.PORT = 4021
001974 CNMSTYLE 2223 RTT.SOCKETS = 50
001975
001976 Statements for function: TCP/IP Connection Data Collector
001977
001978 Member Line# Indicators Statement
001979 -----
001980
001981 CNMSTYLE 751 TOWER = *SA *AON *MSM *Graphics NPDA NLDM
001982 | TCPIPCOLLECT *AMI *TARA *DVIPA *TEMA *IP
001983 | *NVSOA DISCOVERY *ACTIVEACTIVE
001984 CNMSTGEN 112 **** TOWER = *SA *AON *MSM *Graphics NPDA NLDM
1-77
```

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide displays the second section of the report. An ISPF search has been made for **Statements for function**. The components or towers are listed here. The page heading changes for each of the components that are included in the report.

CNMSTYLE report: Third section

Tivoli software

IBM

CNMSTYLE report: Third section

The third section of the report lists the auxInitCmd statements and the user-defined statements

- The auxInitCmd statements display in the order that they are encountered in CNMSTYLE and its included members
- For example, you can define an autotask named OPAAA01 as follows:

```
%> IF TOWER('NPDA') THEN DO;
      function.autotask.MyAutoOp = OPAAA01
%> END;
```
- In the example, the function.autotask.MyAutoOp statement is a user-defined statement

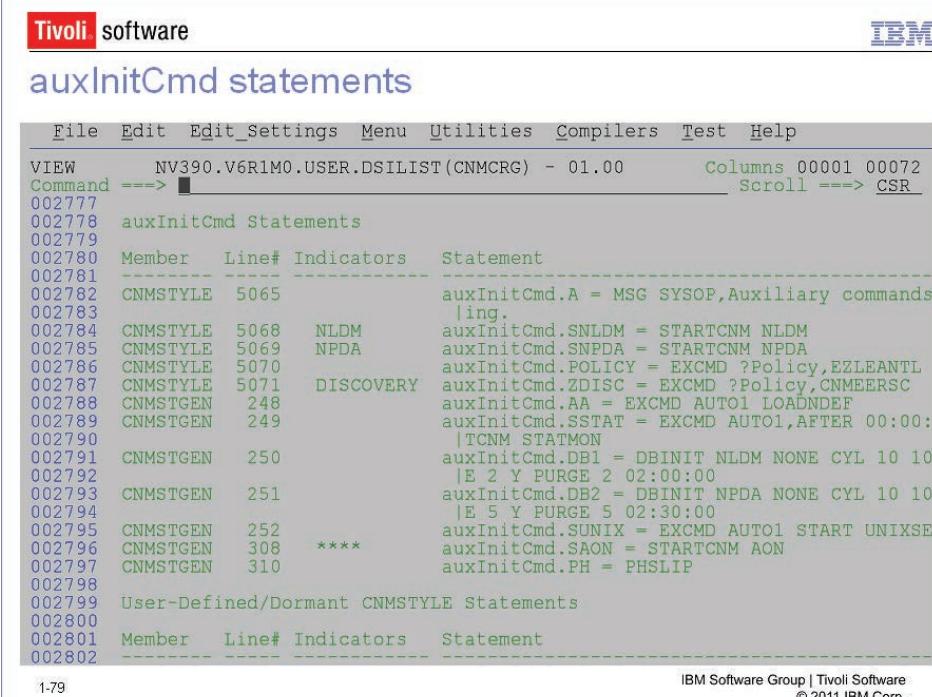
1.78

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The report generator does not recognize user-defined CNMSTYLE statements as belonging to a specific NetView function. These user-defined statements also do not belong to general NetView information in the first section of the report.

auxInitCmd statements

This slide displays the result of a search for auxInitCmd statements, the beginning of the third section of the report. In this example, a message goes to the SYSOP operator indicating that the auxiliary commands are starting. This part of the report also includes the user-defined CNMSTYLE statements.



The screenshot shows a software interface with a menu bar (File, Edit, Edit_Settings, Menu, Utilities, Compilers, Test, Help) and a toolbar (VIEW, Command, Columns, Scroll). The main window title is "auxInitCmd statements". The content area displays a list of statements:

Member	Line#	Indicators	Statement
CNMSTYLE	5065		auxInitCmd.A = MSG SYSOP,Auxiliary commands ing.
CNMSTYLE	5068	NLDM	auxInitCmd.SNLD = STARTCNM NLDM
CNMSTYLE	5069	NPDA	auxInitCmd.SNPDA = STARTCNM NPDA
CNMSTYLE	5070		auxInitCmd.POLICY = EXCMD ?Policy,EZLEANLT
CNMSTYLE	5071	DISCOVERY	auxInitCmd.ZDISC = EXCMD ?Policy,CNMEERSC
CNMSTGEN	248		auxInitCmd.AA = EXCMD AUTO1 LOADNDEF
CNMSTGEN	249		auxInitCmd.SSTAT = EXCMD AUTO1,AFTER 00:00: TCNM STATMON
CNMSTGEN	250		auxInitCmd.DB1 = DBINIT NLDM NONE CYL 10 10 E 2 Y PURGE 2 02:00:00
CNMSTGEN	251		auxInitCmd.DB2 = DBINIT NPDA NONE CYL 10 10 E 5 Y PURGE 5 02:30:00
CNMSTGEN	252		auxInitCmd.SUNIX = EXCMD AUTO1 START UNIXSE
CNMSTGEN	308	****	auxInitCmd.SAON = STARTCNM AON
CNMSTGEN	310		auxInitCmd.PH = PHSLIP
User-Defined/Dormant CNMSTYLE Statements			
002800			
002801	Member	Line#	Indicators Statement
002802			

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

CNMSTYLE report: Fourth section

Tivoli software

IBM

CNMSTYLE report: Fourth section

The fourth section of the report lists Data REXX statements

- These statements display in the report in the order that they are encountered in CNMSTYLE and its included members
- Only the first 63 characters of each Data REXX statement are in the report
- CNMSTYLE statements within a %DATA portion of a Data REXX block that are affected by an IF-THEN statement are also displayed

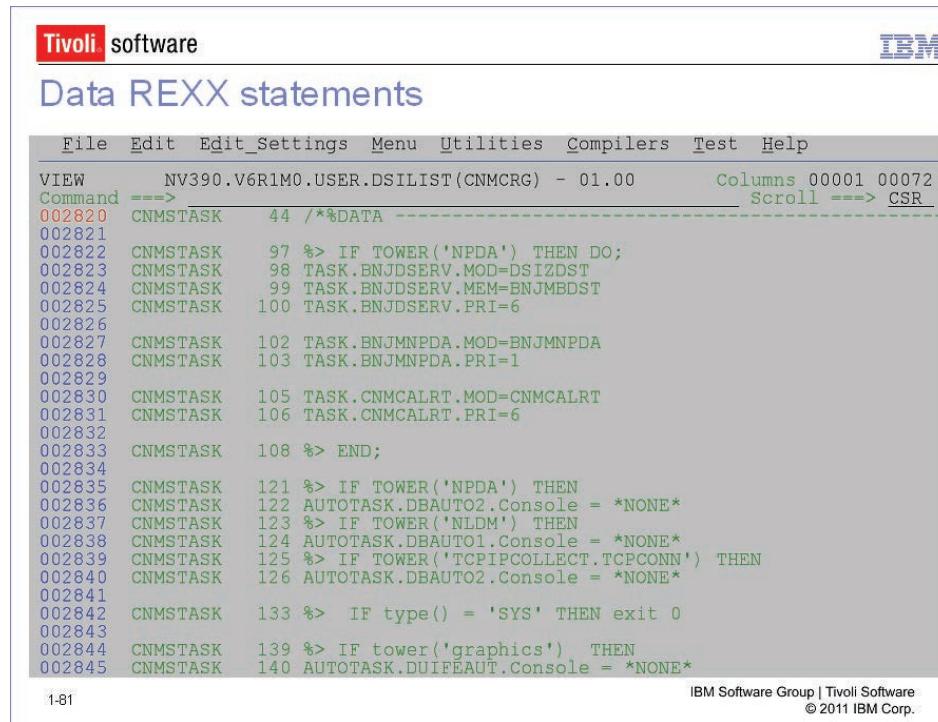
1-80

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The report lists the CNMSTYLE statements with a %DATA statement. This list shows the CNMSTYLE statements that are impacted by your Data REXX statements.

Some %DATA statements that are found are analyzed, and all statements are considered active in the report. Some conditional Data REXX statements are understood in a %DATA portion of code. In general, however, Data REXX statements are not processed. Nothing in a %LOGIC statement is looked at, but all Data REXX show up in the fourth section.

Data REXX statements



The screenshot shows a window titled "Data REXX statements". The menu bar includes File, Edit, Edit_Settings, Menu, Utilities, Compilers, Test, and Help. The main area displays a list of REXX commands. The commands are as follows:

```
VIEW NV390.V6R1M0.USER.DSILIST(CNMCRG) - 01.00
Command ==> CNMSTASK 44 /*%DATA
002821
002822 CNMSTASK 97 %> IF TOWER('NPDA') THEN DO;
002823 CNMSTASK 98 TASK.BNJDSERV.MOD=DSIZDST
002824 CNMSTASK 99 TASK.BNJDSERV.MEM=ENJMBDST
002825 CNMSTASK 100 TASK.BNJDSERV.PRI=6
002826
002827 CNMSTASK 102 TASK.BNJMNPD.A.MOD=BNJMNPD.A
002828 CNMSTASK 103 TASK.BNJMNPD.A.PRI=1
002829
002830 CNMSTASK 105 TASK.CNMCALRT.MOD=CNMCALRT
002831 CNMSTASK 106 TASK.CNMCALRT.PRI=6
002832
002833 CNMSTASK 108 %> END;
002834
002835 CNMSTASK 121 %> IF TOWER('NPDA') THEN
002836 CNMSTASK 122 AUTOTASK.DBAUTO2.Console = *NONE*
002837 CNMSTASK 123 %> IF TOWER('NLDM') THEN
002838 CNMSTASK 124 AUTOTASK.DBAUTO1.Console = *NONE*
002839 CNMSTASK 125 %> IF TOWER('TCP/IPCOLLECT.TCPCONN') THEN
002840 CNMSTASK 126 AUTOTASK.DBAUTO2.Console = *NONE*
002841
002842 CNMSTASK 133 %> IF type() = 'SYS' THEN exit 0
002843
002844 CNMSTASK 139 %> IF tower('graphics') THEN
002845 CNMSTASK 140 AUTOTASK.DUIPEAUT.Console = *NONE*
```

At the bottom left is the page number 1-81, and at the bottom right is the copyright information: IBM Software Group | Tivoli Software © 2011 IBM Corp.

You can find the fourth and final section of the report with ISPF BROWSE by searching for **Data REXX Statements**.

A /*%DATA statement begins the Data REXX group. The conditional %> IF statement determines if the **network problem determination application (NPDA)** tower is enabled. If the NPDA is enabled, it begins a DO group. Within this group, some task statements are assigned. The %> END statement ends the DO group.

Student exercise

Tivoli software

IBM

Student exercise



1-82

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 1-3.

Lesson 8: NetView administration

Defining NetView operators

Tivoli software

IBM

Defining NetView operators

- If SAF security is unused, the OPERATOR statement defines the operators
- DSIOPF is the member that contains the OPERATOR statement
- OPERATOR statements define as follows:
 - Each operator who can log on to this NetView program
 - Each operator who can start a session with this NetView program from a NetView program in another domain
 - Operator identifiers that the AUTOTASK command can start as automation tasks
- You can dynamically add, delete, or change operators as follows:
 - Edit OPERATOR statements
 - Issue the REFRESH OPERS command

1-84

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

There are two ways to enable SAF:

- Code SECOPTS.OPERSEC=SADFEF in CNMSTYLE.
- Specify OPERSEC=SAFDEF on the REFRESH command.

The OPERATOR statement identifies operators who can perform the following tasks:

- Log on to the NetView program.
- Start a session with this NetView program from a NetView program in another domain.

Defining NetView operator profiles (1 of 3)

Tivoli software

IBM

Defining NetView operator profiles (1 of 3)

- PROFILE statement defines the profile name to NetView
- PROFILE must be the first statement in each profile definition
- You code this statement in a member specified by a PROFILEN statement that is associated with the operator
- Profiles are unused when OPERSEC=SAFDEF
- NetView supplies sample profiles named DSIPROFA and DSIPROFB

1-85

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

With OPERATOR defined in DSIOPF and PROFILE defined in a DSIPROFx member, a statement is necessary for connecting the OPERATOR to the PROFILE.

Defining NetView operator profiles (2 of 3)

Tivoli software

IBM

Defining NetView operator profiles (2 of 3)

- PROFILEN statement associates the name of a particular profile or list of profiles with an operator identification
- You code PROFILEN as often as necessary to ensure that all the possible profile names are associated with a particular operator identification
- An OPERATOR statement must precede each PROFILEN statement or group of statements. You code this statement in DSIOPF
- Profiles are unused when SECOPTS.OPERSEC=SAFDEF is specified in CNMSTYLE

1-86

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide describes the PROFILEN statement, which connects the OPERATOR to the PROFILE. The statement follows an OPERATOR definition in DSIOPF. PROFILEN specifies the member where the profile is.

Defining NetView operator profiles (3 of 3)

Tivoli software

IBM

Defining NetView operator profiles (3 of 3)

- Default profile introduced in NetView for z/OS 6.1
- CNMSTYLE definition
DEFAULTS.LogProf = DLPNAME
- The DEFAULTS command is also available at any time, including
DEFAULTS LOGPROF=*NONE*
- The profile itself is defined in DSIPRF, as before
- Operators who use a default profile do not need PROFILEN statements

1-87

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Earlier, SAFCHECK and SAFPW customers needed to define NetView operators in both SAF (e.g. RACF) for passwords, and NetView/DSIOPF for profiles. Also, NETVPW customers needed to specify at least 1 profile for each operator. Now, SAFCHECK and SAFPW customers need not define operators in DSIOPF. NETVPW customers now need not define a profile for each operator.

A new LOGPROF option on the DEFAULTS command must be in effect to enable the DLP before the target logon. A DEFAULTS.LogProf statement is therefore recommended in CNMSTYLE (if DLP is needed).

Example of DSIOPF defining operators

Tivoli software IBM

Example of DSIOPF defining operators

```
NETVIEW.BRWS ----- BROWSE DSIOPF  (DSIPARM ) --- LINE 00051 TO 00068 OF 00555
SCROLL ==> CSR
-----+-----+-----+-----+-----+-----+-----+-----+
1      2      3      4      5      6      7
LABUSR0   OPERATOR    PASSWORD=LABUSR0
          PROFILEN   DSIPROFA
LABUSR1   OPERATOR    PASSWORD=LABUSR1
          PROFILEN   DSIPROFA
LABUSR2   OPERATOR    PASSWORD=LABUSR2
          PROFILEN   DSIPROFA
LABUSR3   OPERATOR    PASSWORD=LABUSR3
          PROFILEN   DSIPROFA
LABUSR4   OPERATOR    PASSWORD=LABUSR4
          PROFILEN   DSIPROFA
LABUSR5   OPERATOR    PASSWORD=LABUSR5
          PROFILEN   DSIPROFA
LABUSR6   OPERATOR    PASSWORD=LABUSR6
          PROFILEN   DSIPROFA
LABUSR7   OPERATOR    PASSWORD=LABUSR7
          PROFILEN   DSIPROFA
LABUSR8   OPERATOR    PASSWORD=LABUSR8
          PROFILEN   DSIPROFA
          OPERATOR    PASSWORD=LABUSR8
          PROFILEN   DSIPROFA
CMD==> ■
TO SEE YOUR KEY SETTINGS, ENTER 'DISPK'
MB a
23/009
```

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This example is a partial display of the DSIOPF member in the DSIPARM data set. It illustrates how an operator statement defines operators named LABUSR0, LABUSR1, and more. The passwords are defined for each operator.

You can associate profiles with the operator by specifying the profile member in DSIPARM to use by using the PROFILEN statement. In this example, each of these operators has been specified to use DSIPROFA.

Example of a PROFILEN definition

The screenshot shows a Tivoli software interface with a red 'Tivoli' logo at the top left. At the top right is the 'IBM' logo. The main window title is 'Example of a PROFILEN definition'. The content of the window is a NetView BROWSE session of the DSIPROFA member. The session output is as follows:

```
NETVIEW.BRWS ----- BROWSE DSIPROFA (DSIPRF ) --- LINE 00000 TO 00016 OF 00016
SCROLL ==> CSR
-----1-----2-----3-----4-----5-----6-----7-----
*****TOP OF DATA***** DATASET: 5
*****
* Licensed Materials - Property of IBM *
* 5697-ENV (C) Copyright IBM Corp. 1986, 2007 *
* All rights reserved. *
* US Government Users Restricted Rights - Use, duplication or *
* disclosure restricted by GSA ADP Schedule Contract with IBM Corp. *
*****
*   NAME(DSIPROFA) SAMPLE(DSIPROFA) RELATED-TO() *
*   DESCRIPTION: NETVIEW OPERATOR PROFILE DEFINITIONS
*****
* MINIMAL NETVIEW OPERATOR PROFILE STATEMENTS *
* WITH INITIAL CLIST AND ONLY UNRESTRICTED COMMAND USE. *
*****
DSIPROFA PROFILE IC=LOGPROF1
AUTH MSGRECVR=NO,CTL=GLOBAL
END
***** BOTTOM OF DATA *****

CMD==> █
TO SEE YOUR KEY SETTINGS, ENTER 'DISPFK'
MB a
23/009
```

At the bottom left of the window, there is a small status bar with '1-89'. At the bottom right, it says 'IBM Software Group | Tivoli Software © 2011 IBM Corp.'

This slide displays the DSIPROFA member. The profile contains the IC keyword, which is used for starting an initial command when the operator has logged on. You can also use an authority statement. When SAFDEF is used, NetView profiles are ignored. The IC parameter must be specified in the NETVIEW segment of the SAF.

DSIPARM member CNMSCAT2

Tivoli software

IBM

DSIPARM member CNMSCAT2

- You use it to group operators according to their responsibilities and roles
- Each operator can connect to as many groups as necessary
- Groups can be defined to allow access to any commands
- Groups can have access restrictions to the following commands:
 - Commands such as NETCONV and NACMD
 - Specific commands that are designed to come from the Tivoli Enterprise Portal
- PROTECT statements specify restricted commands and read or write access to some data sets
- PERMIT statements allow specific operator groups access to specific commands that the PROTECT statements restrict

1-90

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

CNMSCAT2 includes some example statements as follows:

```
GROUP    NVOPS1    NETOP1,NETOP2,OPER1,OPER2,OPER3,AUTO1,AUTO2
PROTECT          *.*.AUTOTASK
PERMIT    NVOPS1    *.*.AUTOTASK
```

SAF administration with CNMSAF2

Tivoli software

IBM

SAF administration with CNMSAF2

- Batch job is in sample member CNMSAF2
- You use CMNSAF2 to set RACF definitions for NetView operators and commands
- Requires modifications before it is usable
- Contains sample statements you can modify as follows:
 - Adding class for the NetView application and commands
 - Adding users (NetView operators)
 - Adding autotask users
 - Specifying initial command for users
 - Adding groups and connecting users to the groups
 - Adding groups that are permitted to issue Tivoli Enterprise Portal agent or EMA commands
 - Permitting groups to log on to specific NetView domains
 - Defining commands that are restricted
 - Permitting groups to access restricted commands
 - Defining TSO user IDs that the NetView operators are surrogates for

1.91

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Some examples of RACF statements in CNMSAF2 are as follows:

```
ADDUSER AUTO1
ALTUSER AUTO1      NETVIEW(MSGRECV( NO ) CTL(GLOBAL) )

ADDUSER NETOP1

ALTUSER NETOP1 NETVIEW(IC(LOGPROF1) MSGRECV(YES) CTL(GLOBAL) )

ALTUSER NETOP1      NETVIEW(NGMFADMN(YES) )

ADDGROUP NVOPS1
CONNECT AUTO1      GROUP(NVOPS1) UACC(READ)
PERMIT domain_name CLASS(APPL) ID(NVOPS1) ACCESS(READ)

RDEF NETCMDS *.*.AUTOTASK UACC(NONE)
PE *.*.AUTOTASK CLASS(NETCMDS) ID(NVOPS1) ACCESS(READ)

RDEF SURROGAT tso-userid.DSITSOSV UACC(NONE)
PE tso-userid.DSITSOSV CLASS(SURROGAT) ID(AUTO1)ACCESS READ)
```

Lesson 9: NetView IP management

Management of SNA over IP

Tivoli software

IBM

Management of SNA over IP

- SNA traffic is transportable over an IP network
- Some useful commands are as follows:
 - DIS: The status of system resources is displayed
 - SESS: A session list panel displays sessions, if the NLDM tower is enabled

1-93

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The NetView support for SNA over IP is specifically for sessions that use the Communications Server function, Enterprise Extender (EE). NetView does not support other methods of running SNA sessions over IP.

Operators can use the Communications Server DIS EEDIAG command and NetView TRACERTE command to perform analysis for these resources. The operators can browse the data to find congestion or broken links.

Dynamic virtual IP address

Tivoli software

IBM

Dynamic virtual IP address

- With Dynamic Virtual IP Address (DVIPA), you can assign a specific virtual IP address to an application
- This virtual address is independent of any specific TCP/IP stack within the sysplex
- If an application must move to another z/OS system because of failure or maintenance, the application is accessible under the same virtual IP address
- Usage of DVIPA is a flexible way to be prepared for failure of application or system
- TCP/IP parameters control DVIPA

1.94

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

For more information about establishing DVIPA addresses, refer to the TCP/IP publications. As an example, for z/OS 1.12, perform the following steps:

1. Navigate to www-03.ibm.com/systems/z/os/zos/bkserv/r12pdf/.
2. In the contents, click the Communications Server link.
3. Open the *z/OS VIR12.0 Communications Server IP Configuration Guide*. (Part 1 Chapter 7 is titled “Virtual IP addressing.”)

Support for DVIPA

Tivoli software

IBM

Support for DVIPA

- DVIPA information availability through the following user interfaces:
 - Tivoli Enterprise Portal
 - 3270 console
- DVIPA definition and status information, including the differentiation of application-instance, stack-defined, and distributed DVIPAs
- Distributed DVIPA information, including sysplex distributors, distributed targets, application server health statistics for distributed targets, and statistics on workload balancing
- DVIPA connection information, including the number of active connections and information about the current state of the connection
- DVIPA routing information, including VIPA routes and distributed DVIPA connecting routing
- Historical DVIPA information

1-95

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Through the NetView interfaces for 3270, and Tivoli Enterprise Portal, you can see DVIPA information in various displays. For Open Systems Adapter (OSA) trace collection, z/OS V1R11 is a prerequisite.

Dynamic IP resource discovery

Tivoli software

IBM

Dynamic IP resource discovery

- The MSM topology manager dynamically discovers the topology and status of IP resources in the network
 - Includes resources that runs in z/OS and distributed environments
 - Stores the information in the Resource Object Data Manager (RODM)
 - After the information is in RODM, you can view your network resources from the NetView Management Console
- Topology correlation information is as follows
 - Automatically correlates resources that different types of topology functions manage, such as IP and Tivoli management region
 - Available for the following items:
 - MultiSystem Manager topology functions
 - NetView SNA Topology Manager
 - Customer or vendor applications that use the Graphic Monitor Facility host subsystem data model

1.96

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Using the NMC user interface, you can view the topology of the network.

IP connection management

Tivoli software



IP connection management

- The NetView program provides both real-time and historical connection information, including the following items:
 - Stack name
 - Local and remote addresses and ports
 - Start time
 - End time (for connections that ended)
 - Termination code (for connections that ended)
 - Sent and received byte and segment counts
 - Retransmit counts
 - Information about connection state
 - Interface
 - Host
 - TN3270
 - Application transparent transport layer security (AT-TLS), if applicable
- You can list this data with the TCPCONN and CNMSTCPC commands, and view the data using the Tivoli Enterprise Portal
- Data is available as both a readable form and as binary form for programming use
- Supports host name translation and IPv4 or IPv6 addresses
- You can use cross-domain capabilities of the NetView program for viewing the connection data at remote z/OS hosts

1-97

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

CNMSTCPC is intended as an operator interface. It is a sample CLIST that formats output for readability. Because it is a sample, you can customize the display. TCPCONN is intended as a programming interface. TCP/IP connections are viewable in these interfaces:

- 3270
- Tivoli Enterprise Portal

Packet trace collection and formatting

Tivoli software

IBM

Packet trace collection and formatting

- The NetView program provides real-time capture and formatting of IP packet trace data, including both headers and payloads
- Because the formatter is directly integrated with the IP stack, no translation mismatches occurs
- Highly flexible tracing and formatting options are available so you can filter out unwanted data
- Both IPv4 and IPv6 packets are supported. The data is also available in binary (unformatted) form for use by automation routines

1.98

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The formatting of packet trace data is the same as for under IPCS. You do not need to learn a new format. The PKTS command is intended for programming purposes. The FMTPACKT command is intended for operators. Packet trace data is viewable using the 3270 interface. From the Tivoli Enterprise Portal interface, issue a NetView PKTS or FMTPACKT command.

OSA trace collection and formatting

Tivoli software



OSA trace collection and formatting

Examining packet content is sometimes necessary to debug a problem

- Supports tracing of OSA packets with OSA-Express2 Network Traffic Analyzer (OSAENTA)
- You can capture the following items:
 - Ethernet data (Ethernet type, source/destination MAC addresses, VLAN tag, LLC fields)
 - IPv4 & IPv6 data
 - ARP packets
 - SNA transmission headers
 - Direction indicators
 - Discard code
 - Interface identification
- Syntax and operation are similar to packet trace function

1-99

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

For Open Systems Adapter (OSA) trace collection, z/OS V1R11 is a prerequisite. OSAENTA is supported by only the OSA-Express2 feature, which requires OSA Express2 card or higher. OSA Express does not support OSAENTA. The following types of data are collected:

- Data from the Ethernet header as follows (when present):
 - Source and destination MAC addresses
 - Ethernet type
 - VLAN tag (when present)
 - Link Layer Control (LLC) fields (when present)
- IPv4 and IPv6 protocol headers
- ARP packets
- SNA transmission header
- A device identifier that identifies the interface that the packet is flowing through
- A direction flag
- A discard code, such as a non-zero value, that identifies the reason for discarding the packet.

Commands are PKTS and FMTPACKT, the same as for TCPIP tracing but with other options.

Formatted trace

Tivoli software

IBM

Formatted trace

IPTTRACE command

- Manages IP packet traces
- Displays packet trace data

FKXXK2A01 IPTTrace Control Center			AOFDA	
Service Point/Stack: ADCCD1		Proc: TCPPIP	Domain: LOCAL	
		Status/Owner	Start	For Writer
█	CTRACE	SYSTCPPIP	ACTIVE/NA	NA *NONE*
—	PKTTRACE	SYSTCPDA	NONE/NA	NA *NONE*
—	OSATRACE	SYSTCPOT	ACTIVE/NA	NA *NONE*

Command ---->

1-100

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The IPTTRACE command is used to manage tracing and formatting. IPTTRACE uses PKTS, FMTPACKT.

Command support

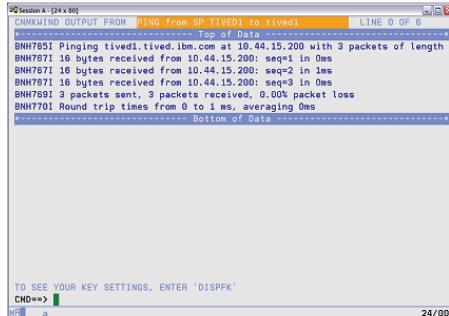
Tivoli software

IBM

Command support

- Commands can be issued directly from the NetView command line, in REXX procedures, and other automation routines
- Commonly used commands are as follows:
 - PING: Test connectivity to an IP host
 - TRACERTE: From the IP stack on the host where the NetView program runs, trace the routes of data packets to a specified IP host. Determine connectivity, roundtrip times, and routers
 - TN3270: Log on to remote TCP/IP-connected systems, either from the NetView command line or from the NetView management console
 - SNMP commands: Send an SNMP request to a network device to set or obtain information about the device
 - SOCKET: Request IP services to obtain information about the TCP/IP stack being used, or to manage client or server applications (or both)
 - REXEC: Send a command over IP to a remote host for processing by using UNIX RSH protocol. The resulting output is displayed
 - RSH: Send a command over IP to a remote host for processing

1-101



The screenshot shows a window titled "Screen A (12x4x60) ENHANCED OUTPUT FROM PING: from SP TIVEDI to tived1". The window displays the following text:
BNH765I Pinging tived1.tived.ibm.com at 10.44.15.200 with 3 packets of length 1
BNH767I 16 bytes received from 10.44.15.200: seq=1 in 0ms
BNH767I 16 bytes received from 10.44.15.200: seq=2 in 1ms
BNH767I 16 bytes received from 10.44.15.200: seq=3 in 0ms
BNH769I 3 packets sent, 3 packets received, 0.00% packet loss
BNH770I Round trip times from 0 to 1 ms, averaging 0ms

Bottom of Data

TO SEE YOUR KEY SETTINGS, ENTER 'DISPFP'
CHD=2 24/008

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The example shows the results from a PING command, one of the most commonly used IP commands.

REXEC is available from the command line or in a pipe. REXEC client support is provided for execution of functions in remote systems. REXEC server support executes a command that is sent from a remote host. The remote host must have an REXEC server monitoring at the specified or default port for this command to work.

Remote Shell (RSH) client support starts a UNIX shell on a remote system and runs a single command in that shell. RSH ends on the remote system when the command ends. RSH server support executes a command that is sent from a remote host to NetView. This command is also available from the command line or in a pipe. For RSH, the output can be displayed as line-mode output or in a panel that is placed on the NetView roll stack. If the remote host supports it, additional commands can be issued from the panel where the output displays.

Automated responses to intrusions

Tivoli software

IBM

Automated responses to intrusions

- Within a firewall, systems can be vulnerable to attack or misuse, whether accidental or malicious, referred to as intrusions
- In conjunction with the Intrusion Detection Services (IDS) of z/OS Communications Server, the NetView program offers several kinds of automated responses to intrusions
 - Notifications: Send the following types:
 - An email to security administrators
 - an alert to the NetView console
 - A message to designated NetView operators
 - A Tivoli Enterprise Console event to Tivoli Risk Manager for enterprise-wide correlation and analysis
 - Commands: Issue UNIX, NetView, or z/OS commands to collect more data, or take other actions
 - Statistics: Collect statistics and generate trmdstat reports to send to security administrators by email

1-102

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

SNMP support for commands

Tivoli software

IBM

SNMP support for commands

- The NetView SNMP command sends an SNMP request to a network device to set or retrieve information about the device
- The SMNP command can be used for the following purposes:
 - Retrieve management information base (MIB) variable values
 - Set MIB values
 - And more

1-103

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The SNMP command is used for retrieving and altering **Management Information Base (MIB)** information. Different versions of SNMP are supported by the SNMP command. Valid versions are 1, 2c, and 3, with the default of 1. The SNMP functions are as follows:

- SNMP BULKWALK: Walk a subtree of a specified MIB variable.
- SNMP GET: Retrieve MIB variable values.
- SNMP GETBULK: Retrieve values for MIB variables in a branch or single leaf node.
- SNMP GETNEXT: Retrieve the value of the MIB variable following the specified MIB variable.
- SNMP INFORM: Send an INFORM request PDU to an SNMP agent or manager.
- SNMP SET: Set MIB variable values.
- SNMP TRAP: Send a TRAP PDU to an SNMP agent.
- SNMP WALK: Retrieve values of all MIB variables in a branch.

The following SNMP example sends a TRAP **protocol data unit (PDU)** request:

```
snmp trap -v2c -p 2005 -c public tvt2010 99
    1.3.6 .1.3.6.1.2.1.1.6.0 s 'this is a trap2_pdu'
```

Network address translation

Tivoli software

IBM

Network address translation

- IP addresses can change as packets route from one network to another
- For easy recognition, NetView uses a special symbol to flag addresses that are translated
- NetView graphical displays provide the corresponding original address

1-104

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

IPv6 support and enablement

Tivoli software



IPv6 support and enablement

- IPv6 is enabled for NetView commands, services, and components
- IPv6 addresses are enabled as input for commands
- IPv6 addresses can be displayed in most places where an IP address are viewable as follows:
 - Messages
 - Views
- You can operate NetView in an IPv6, IPv4, or mixed environment seamlessly because both IPv4 and IPv6 addresses are recognized as valid IP addresses
- With new environment variable *IPv6Env*, users can isolate NetView from one IP version to another

1-105

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

NetView supports IPv6 as follows:

- Provides seamless enablement of IPv6 addresses as input to commands and in output.
- Supports the operation of NetView and its components in an IPv6-enabled network environment.
- Enables users to isolate NetView from either IP version (IPv4 or IPv6) by using an environment variable, *IPv6Env*, which is defined in the style sheet.

IP host names are resolved to either an IPv4 or IPv6, or whichever comes back first, depending on the *IPv6Env* environment variable setting. Minimal changes to some AON IP management panel displays have been made. Longer IPv6 addresses can now be accommodated. Few messages and other output displays are fundamentally altered.

Summary

Tivoli software



Summary

Now that you have completed this unit, you can perform the following tasks:

- Describe the packaging and installation of NetView
- Describe structure and components of NetView
- Use JCL procedures and parameters that are necessary to run NetView
- Use the NetView 3270 interface
- Customize NetView using CNMSTYLE
- Use the CNMSTYLE report generator
- Describe NetView operator administration
- Describe IP management with NetView

1-106

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Unit 2: NetView user interfaces and product integration

Tivoli software

IBM

Unit 2: NetView user interfaces and product integration



© 2011 IBM Corp.

Introduction

This unit presents the non-3270 interfaces for accessing NetView functions: Tivoli Enterprise Portal, web application, and NMC. This unit also contains information about product integration with IBM Tivoli OMEGAMON XE products.

Objectives

Tivoli software

IBM

Objectives

When you complete this unit, you can perform the following tasks:

- Describe the NetView web interface
- Describe the NetView web services gateway interface
- Describe the NetView Management Console (NMC) interface
- Use the Tivoli Enterprise Portal and the NetView EMA workspaces
- Explain how the NetView product integrates with IBM Tivoli OMEGAMON XE products

Lesson 1: NetView web interface

NetView web application

Tivoli software **IBM**

NetView web application

- Provides web access to NetView
 - Enter the web address of the NetView Web server, which includes the destination NetView domain identifier (ID)
 - Type a valid NetView operator ID and password
- Requires one of the following servers:
 - IBM WebSphere Application Server
 - Embedded version of IBM WebSphere Application Server – Express
- From NetView for z/OS 6.1, the web application is minimal, with no NetView functions available



2-4

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide shows a logon screen for the NetView Web application, with prompts for a user name and password. After you enter the NetView web address, the logon window opens. Type the user name and password, and click **OK** to enter the application.

Lesson 2: NetView web services gateway

NetView web services gateway features

Tivoli software

IBM

NetView web services gateway features

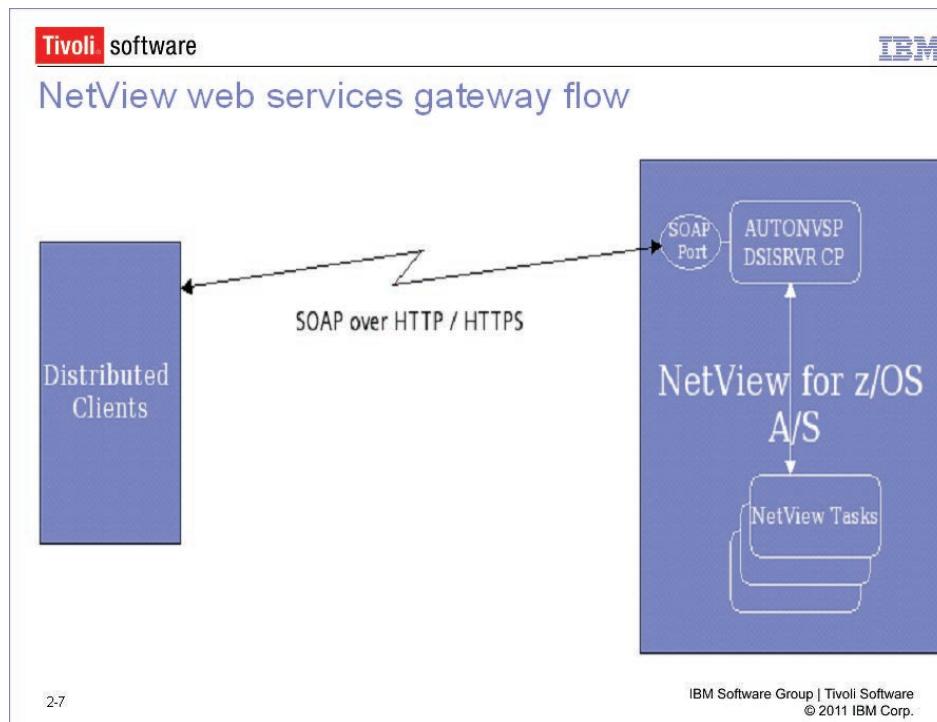
- Provides an industry-standard open interface into the NetView program
- Allows distributed applications (whether written by IBM or customer) for interacting with NetView
- Provides services that are independent of platform, environment, application language, or programming model
- Implemented as SOAP server
- Different types of client applications (such as Java, Microsoft .NET, and third-party applications) can submit SOAP requests to NetView to extract data
- Does not require WebSphere or any other middleware
- Documentation: *IBM Tivoli NetView for z/OS 6.1 Application Programmer's Guide*

2-6

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You access data through the gateway. Anything that NetView can access or store, such as RODM, TCP/IP, Sysplex, is available. All data is text-based.

NetView web services gateway flow



CNMSTYLE is used for setting up the SOA server, which is IPv6-enabled. The server supports SSL user cache, Cert Auth, and different cipher suites. You can perform the following tasks with the NetView web services gateway:

- Run all NetView line-mode commands.
- Provide automation for external messages.
- Provide both secure and non-secure communication.
- Use the product-supplied file, Web Services Description Language (WSDL), for generating static or dynamic proxy clients.
- Customize the output by using CNMSTYLE.
- Use product-provided debug tools, such as Trace, Simple Object Access Protocol (SOAP) test client, and other help tools.
- Start multiple server sessions for load balancing, security, or customization.
- Use the gateway as a basic HTTP/HTTPS server.

NetView web services gateway example

The screenshot shows a web browser window titled "Tivoli software" with the URL "http://mvscf01.lisvpn.ibm.com:9998/znvsoatx.htm". The page is titled "NetView web services gateway example". It features a "NetView for z/OS Generic SOAP Client" section. On the left, there's a "Enter your SOAP Request here:" text area containing XML code for a LISTVAR command. On the right, there's a "Your Soap Response Payload:" text area showing the resulting XML response. Below these are sections for "SOAP Request Headers" and "SOAP Response Headers". At the bottom right, it says "IBM Software Group | Tivoli Software © 2011 IBM Corp."

The example reflects outcome from the following actions:

1. Connecting to <http://hostip:9998/znvsoatx.htm>.
2. In the Payload (XML) field, changing the user ID and password to valid ones.

The SOAP Response is for LISTVAR command.

Lesson 3: NetView Management Console (NMC) interface

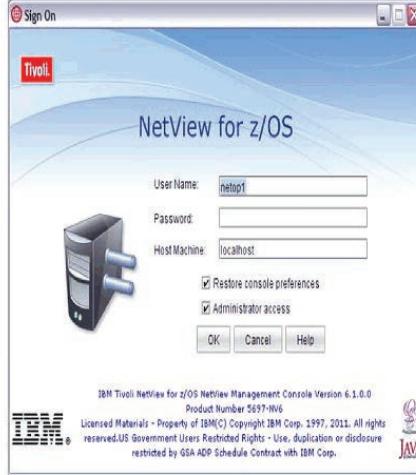
NetView Management Console (NMC)

Tivoli software

IBM

NetView Management Console (NMC)

- The NetView Management Console (NMC) is a workstation-based application
- The NMC displays graphics with depictions as follows:
 - The resources that represent a network
 - A portion of a network
 - A group of networks at various levels of detail
- These views display the network and systems resources that you are monitoring
- When you monitor a network, the resource statuses graphically change in the views



IBM Tivoli NetView for z/OS NetView Management Console Version 6.1.0.0
Product Number 5697-NV6
Licensed Materials - Property of IBM/C Copyright IBM Corp. 1997, 2011. All rights reserved.
US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

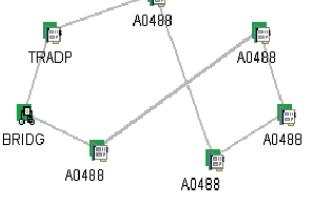
2-10

The NMC displays the resources in the network in several ways, showing the connectivity and topology. When you select the workstation program, a logon screen opens, prompting for a user name, a password, and a host location. The slide shows the NMC logon screen, which is a window on a workstation.

Performing tasks with NMC (1 of 2)

Tivoli software 

Performing tasks with NMC (1 of 2)



- Monitor and control large portions of complex business systems
- View the topology and connectivity of your network graphically
- Monitor the overall state of a network or a portion of a network by using aggregates. Aggregates represent the combined status of a group of related applications and resources
- Navigate from an aggregate to a real resource that is failing
- Mark resources for your own purposes, for example, to show that they are being serviced
- View a list of events or status changes for a selected resource displays
- Issue commands to a resource

2-11

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The graphic on this slide is an example of a *physical configuration view*. This view arranges network resources in a configuration based on a physical relationship between the resources. Commands can be issued against a resource:

- For SNA resources, VTAM commands can be issued.
- For IP resources, IP commands, such as PING and SNMP, can be issued.

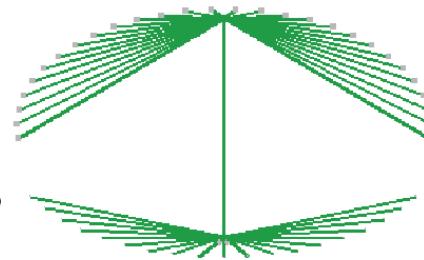
Performing tasks with NMC (2 of 2)

Tivoli software

IBM

Performing tasks with NMC (2 of 2)

- Issue predefined commands from context menus, or issue your own commands
- Issue the same command to multiple NetView domains at one time
- Stop and restart selected resources
- Monitor and manage multiple NetView programs
- At intervals, cycle through open views automatically
- Build custom view and aggregate resource collections
- Monitor resources by exception, to be displayed on the screen only when the resources need the attention of the operator



2-12

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The graphic on this slide is an example of a *configuration backbone view*. This view arranges network resources in a configuration that is based on a subarea backbone relationship.

Lesson 4: NetView EMA

NetView EMA overview

Tivoli software

IBM

NetView EMA overview

- The NetView EMA provides the capability for you to manage your network from the Tivoli Enterprise Portal
- Both sampled and real-time NetView data are available
- With the EMA and OMEGAMON XE performance agents, you can manage and view availability and performance data for your network from a single interface

2-14

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The NetView EMA integrates with IBM Tivoli Monitoring. The NetView EMA workspaces are present in the Tivoli Enterprise Portal Navigator, which is described in another lesson.

Performing tasks with the NetView EMA

Tivoli software

IBM

Performing tasks with the NetView EMA

- Monitor NetView applications
- Monitor NetView task status and performance statistics
- Monitor the status of your TCP/IP stacks
- Monitor DVIPA configuration, workload balance, connections, connection routing, and VIPA routes
- Monitor and diagnose problems with TCP/IP connections
- Monitor the configuration and status of your Telnet servers
- Monitor the configuration and status of OSA channels and ports
- Monitor the configuration and status of HiperSockets™ interfaces
- Monitor active SNA sessions
- Issue commands to manage your network
- Support for the GDPS® Active/Active Continuous Availability solution

2-15

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

NetView EMA and data collectors

Tivoli software



NetView EMA and data collectors

- NetView EMA
 - Provides more integration into Tivoli Enterprise Portal
 - Provides sampled workspaces, including new NetView Health workspaces
 - Provides Take Action commands with improved interface for security
 - Provides situations and expert advice
 - Provides some historical tables
 - Provides cross-product links with multiple OMEGAMON XE agents that use Dynamic Workspace Linking (DWL)
- Data collectors for NetView EMA
 - Contained within the NetView address space
 - Customization done in CNMSTYLE

2-16

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The components of the NetView EMA can run on both the mainframe and distributed environments:

- Mainframe: The collectors collect data from the mainframe and store it in the Tivoli Enterprise Monitoring Server.
- Distributed: The operator accesses data through the Tivoli Enterprise Portal and Tivoli Enterprise Portal Server. The Tivoli Enterprise Monitoring Server provides the data to the operator.

Deployment

Tivoli software

IBM

Deployment

- The NetView EMA data files are on the data files CD of IBM Tivoli NetView Enterprise Management Agent
- The installation process is the *Installation: Configuring the NetView Enterprise Management Agent* manual (See “Chapter 5, Completing the NetView agent configuration”.)

2-17

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Refer to the following sections in the *Installation: Configuring the Tivoli NetView for z/OS Enterprise Agents* manual:

- Windows environment: Follow the instructions in the section titled “Loading files and configuring the Tivoli Enterprise Portal in a Windows environment.”
- Linux or Advanced Interactive Executive (AIX) environment: Follow the instructions in the section titled “Loading files and configuring the Tivoli Enterprise Portal in the Linux and AIX environments.”

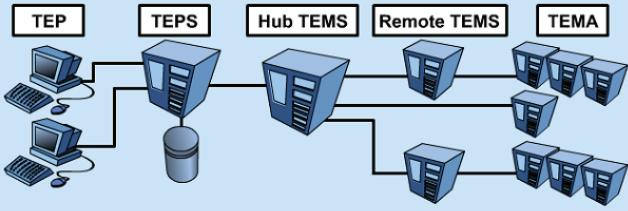
Lesson 5: Tivoli Enterprise Portal

Architecture and components

Tivoli software 

Architecture and components

- The Tivoli Enterprise Portal is part of the IBM Tivoli Monitoring client-server architecture
- This infrastructure includes all components needed in an enterprise monitoring solution
- Those components are called Tivoli Monitoring Services (TMS), which include the following items:
 - Tivoli Enterprise Portal: Graphical user interface, either a desktop application or use of a browser
 - Tivoli Enterprise Portal Server: Repository for all user data
 - Tivoli Enterprise Monitoring Server: Central repository for all monitored data. (This can be on a z/OS system.)
 - Tivoli Enterprise Monitoring Agent: The monitoring data collectors for operating systems and application. The NetView Enterprise Management Agent (EMA) is a Tivoli Enterprise



2-19

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

IBM Tivoli Monitoring can perform the following tasks:

- Monitor and manage operating systems, applications, databases, workloads, networks, and other resources.
- Monitor a variety of platforms, both distributed and mainframe.
- Track availability and performance of the enterprise.
- Establish performance thresholds and raise alerts.
- Monitor resources for certain conditions.
- Provide reports for tracking trends and troubleshooting problems.
- Visualize real-time monitoring data.
- Create and send commands to systems.
- Define custom queries to monitor items of interest.

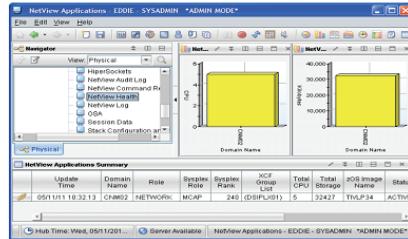
Two Tivoli Enterprise Portal modes: Desktop client and browser

Tivoli software

Two Tivoli Enterprise Portal modes: Desktop client and browser

- The desktop client is installed on a workstation
- An IBM Tivoli Monitoring administrator typically performs the installation
- You start the client by finding and double-clicking the desktop client application icon
- Many users access the Tivoli Enterprise Portal client in browser mode by using the following web address:
hostname:1920///cnp/kdh/lib/cnp.html







IBM Software Group | Tivoli Software
© 2011 IBM Corp.

2-20

For the desktop client, the software installs directly on the workstation that is monitoring the environment. For the browser interface, you can enter the web address for Tivoli Enterprise Portal Server and port number on any device in the network.

With either interface, you first see the logon window. This window prompts you to enter the user ID and password, and click **OK**. Some Java code might download when the Tivoli Enterprise Portal starts. When the Tivoli Enterprise Portal starts, a default screen called a *workspace* opens. In the example, the NetView Health workspace was selected from the Navigator.

A third way to access the Tivoli Enterprise Portal is by using Java webstart (javaws). The java code automatically updates when changes occur to the Tivoli Enterprise Portal Server. To use the Java webstart, issue the following command

```
javaws http://tepsserver:1920///cnp/kdh/lib/tep.jnlp
```

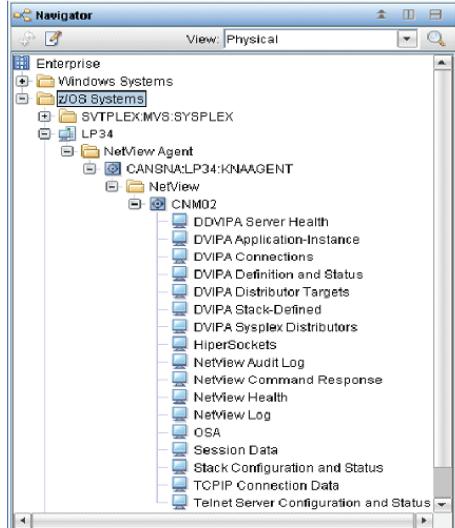
NetView EMA Navigator

Tivoli software

IBM

NetView EMA Navigator

- The Navigator is in the upper left part of the window and contains items that build the different levels in its hierarchy
- The default navigator view is a physical hierarchy of items
- Each navigator item contains one or more workspaces that provide information relevant to that level of the navigator



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

2-21

The NetView Navigator is in the upper left corner of the screen. The Navigator contains a hierarchy of operating system types, operating systems, host names, and agents. For NetView, it can further expand to the NetView domain and NetView-specific workspaces.

The slide shows the Navigator view as a *physical* view. You can display a *logical* view by clicking the **View** list and clicking **Logical**. For selected workspaces, you can customize logical views. The NetView EMA workspaces contain views that report information about enterprise resources that you are monitoring.

Most of the following workspaces contain sampled data as follows:

- DVIPA workspaces
- Transmission Control Protocol/Internet Protocol (TCP/IP) workspaces
- NetView Health workspaces
- Session Data workspaces

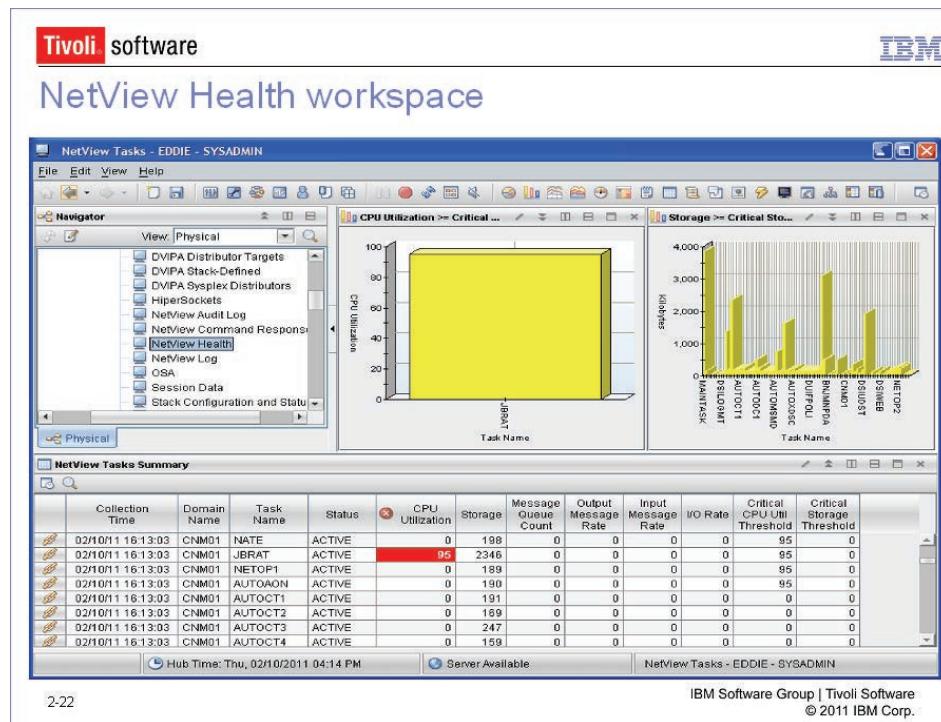
Some of the following workspaces contain real-time data:

- NetView Audit Log workspace
- NetView Command Response workspace
- NetView Log workspace
- Stack Configuration and Status workspace

Use the workspaces to perform the following tasks

- Monitoring NetView task status and performance metrics
- Monitoring the status of your TCP/IP stacks in a sysplex
- Monitoring DVIPA configuration, workload balance, and connections
- Using TCP/IP connections to monitor and diagnose problems
- Monitoring active SNA sessions
- Using packet trace to diagnose TCP/IP problems
- Issuing commands for managing your network

NetView Health workspace



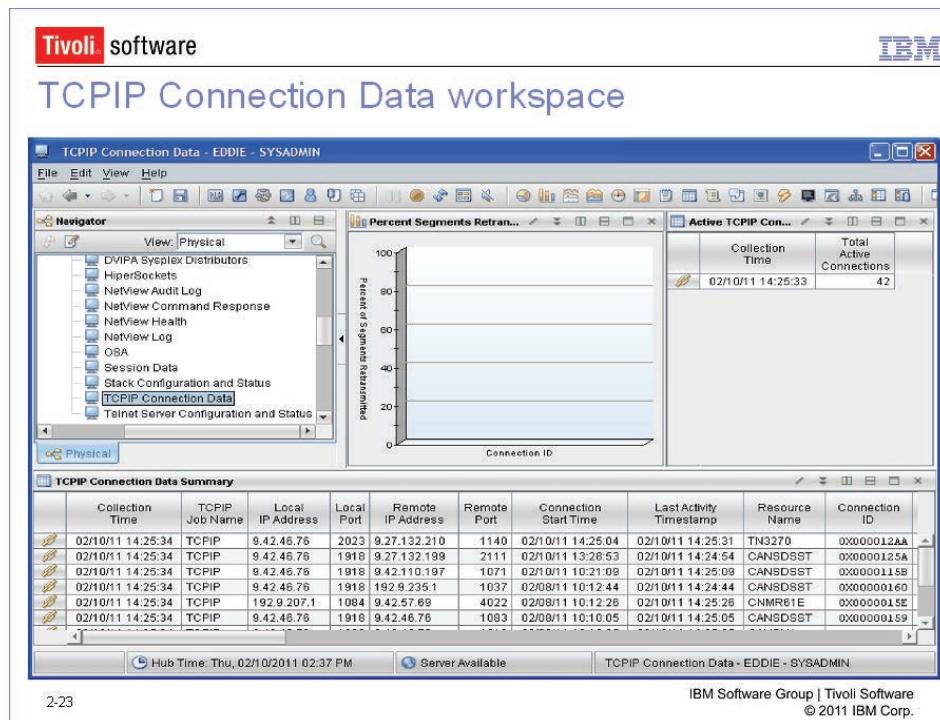
The workspace in the slide example is *NetView Health*. It provides monitoring information about processors, storage, and tasks for NetView.

The view to the immediate right of the Navigator is the *CPU Utilization >= Critical CPU Util Threshold*. It shows the task names and processor utilization for the tasks.

Next to the CPU view is the *Storage >= Critical Storage Threshold*. The bar chart shows the task names and the amount of storage used in kilobytes.

In the view that contains the *NetView Tasks Summary* table of information, each line pertains to a single task. A blue link symbol precedes each line of task data. You can click a link to go directly to a workspace that contains information about that specific task. The second row in the NetView Tasks Summary table has a red field for the CPU usage. In Tivoli Enterprise Portal, you can customize tables so that certain values show up as warnings or errors, making it easier to spot problems.

TCPIP Connection Data workspace



The workspace in the slide example is **TCPIP Connection Data**. It contains these following views:

- **Percent Segments Retransmitted >= 3** lists the percent of segments retransmitted for connection IDs.
- **Active TCP/IP Connection Count** shows the total active connections.
- **Filtered TCPIP Connection Data Summary** contains local and remote IP addresses and ports, resource names, bytes sent and received, and more.

NetView EMA views

Tivoli software

IBM

NetView EMA views

The screenshot shows the NetView Applications - EDDIE - SYSADMIN workspace. It contains three views:

- NetView CPU Utilization:** A bar chart showing CPU utilization for three servers: S0001, S0002, and S0003. The Y-axis ranges from 0.00 to 1.00.
- NetView Storage:** A bar chart showing storage usage for three servers: S0001, S0002, and S0003. The Y-axis ranges from 0,000 to 32,000.
- NetView Applications Summary:** A table listing application details for three domains: CNW01, THW01, and THW02. Columns include Update Time, Domain Name, Role, System Role, Status, Rank, Avg. Overall Util, Total CPU, Total Storage, JES3 Image Name, Status, Network ID, PORTNO, PAddress, Port, and Ver.

2-24

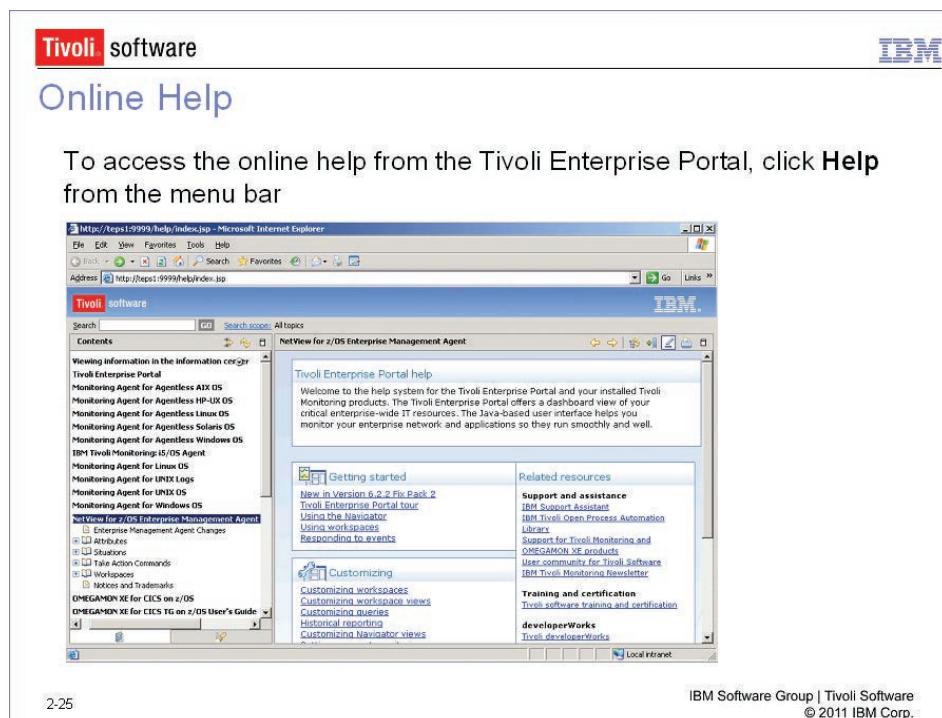
- The application window displays a workspace. In this example, the workspace contains three views: two bar charts and a table
- You use different types of views to display enterprise monitoring data or access to other areas of interest
- Each application (type of agent) comes with a set of predefined workspaces
- Workspaces and views are modifiable to fit the interest or responsibility of the users

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Default views are provided with each workspace. You can modify views as follows:

1. Click a different view type icon from the toolbar while holding down the left mouse button.
2. Move the cursor to the view that you want to modify.
3. Release the mouse button.
4. When a window opens, select attributes for the view.
5. When you leave the modified workspace, respond to the prompt to either discard the workspace or save it under another file name.

Online Help



The NetView EMA Help explains attributes, situations, Take Action commands, and workspaces.

Summary

Tivoli software

IBM

Student exercise



2-26

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform all exercises in Unit 2.

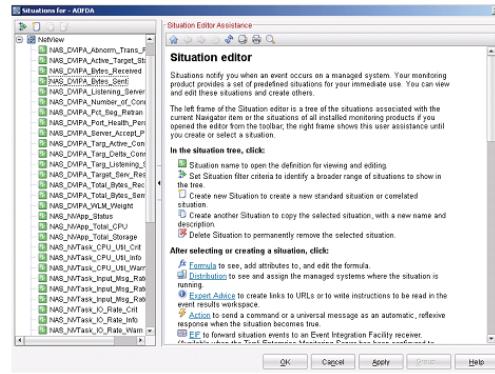
Product-provided situations

Tivoli software

IBM

Product-provided situations

- Situations set up conditions for alerting and possible automated actions
- You create or modify situations by using the Situation editor
- To access the Situation editor, you click the Situation editor icon on the Tivoli Enterprise Portal toolbar
- To view the situations, you expand the NetView pane
 - The situation names must be unique
 - NetView situations begin with the prefix NAS
 - NA is the product code and S indicates situation



2-27

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

In the Situation Editor window, product-provided situations display under the NetView heading in the pane on the left side. The Situation Editor Assistance pane is on the right side. This pane provides help for creating a new situation, creating another situation that is based on a copy of an existing situation, and deleting a situation.

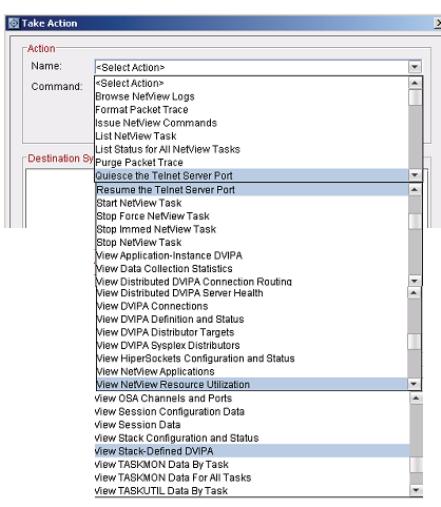
When you create a new situation or create one from an existing situation, a window opens in the right pane of the Situation Editor. In that window, you can define items as follows:

- The situation description
- A formula that calculates in the situation
- The frequency to run the situation
- The systems to run the situation on
- Any expert advice to provide to the operator
- A system command to issue as a Take Action
- The time to stop running the situation

Take Action commands

Tivoli software **IBM**

Take Action commands



- Take Action commands are provided with the NetView EMA so that operators can use Tivoli Enterprise Portal to issue real-time commands to NetView
- Some commands are context-sensitive, which means that they insert a value from a row in a workspace in the command
- The NetView Command Response workspace displays commands and command responses for each Tivoli Enterprise Portal user ID

2-28

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The NetView Audit Log, NetView Command Response, Session Data, and Stack Configuration and Status workspaces provide a *Take Action* view. From this view, you can select a Take Action command, supply parameters, and run the command. The **Issue NetView Commands** action provides you with an area for entering the exact command that you plan to run.

Take Action command execution

Tivoli software

IBM

Take Action command execution

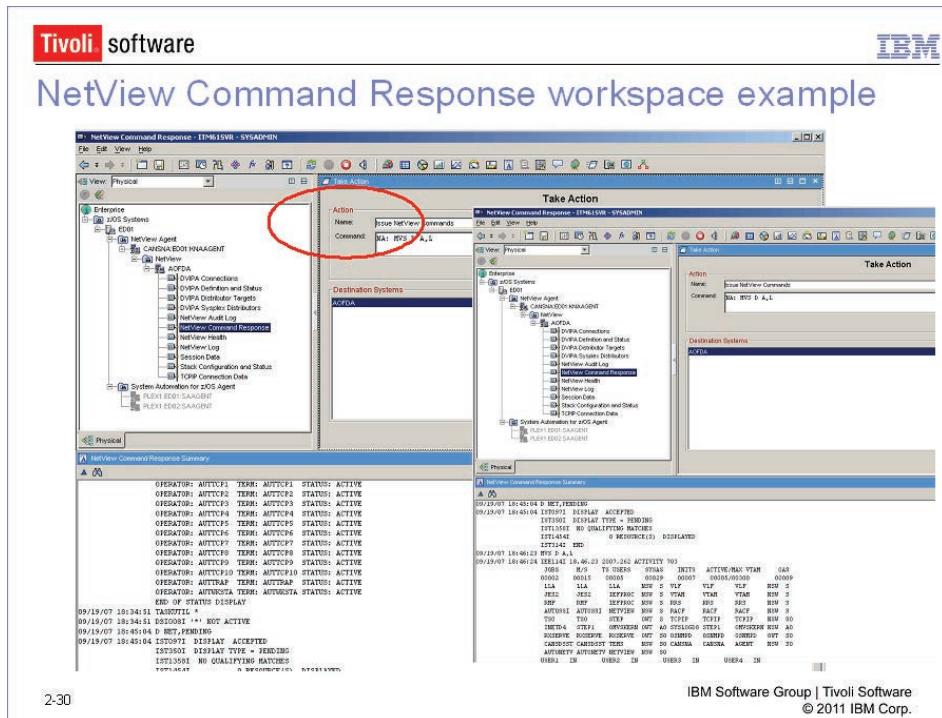
- Take Action commands that route to NetView include the Tivoli Enterprise Portal user ID
- You use the Tivoli Enterprise Portal user ID for determining the NetView operator ID that authorized and ran the command
- The command runs under a NetView ID determined by the following conditions:
 - Validation of the Tivoli Enterprise Portal user ID according to the NetView operator ID criteria. (See *IBM Tivoli NetView for z/OS Administration Reference*.)
 - Mapping of the Tivoli Enterprise Portal user ID to a NetView operator ID in the CNMSTYLE member by using the NACMD.OPID.TEPLogonID variable
 - Use of the Tivoli Enterprise Portal user ID

2-28

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The NetView Command Response workspace contains both a *Take Action* view and a *NetView Command Response Summary* view. The result of the command that you issue is in the command response view.

NetView Command Response workspace example



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

To run a Take Action command, select a NetView domain for the destination, and click the **Run** button. The output from the command displays in the NetView Command Response Summary view. In the example, an MVS D A,L command is issued. The active tasks are listed in the NetView Command Response Summary view.

Historical data collection (1 of 3)

Tivoli software

IBM

Historical data collection (1 of 3)

- Both real-time and historical data are available within the NetView EMA workspaces
- After historical data is configured, enabled, and collected, historical reports can display information
- Historical reports are useful for two significant reasons:
 - Finding the root cause of problems that evolved over a period of time
 - Debugging problems that occurred in a previous time period
- Capacity planners can also use historical reports to identify trends and correct imbalances in network load distribution
- To generate reports containing historical data, historical collection must be configured and enabled ,and data must be collected
 - Click the History Configuration icon in the Tivoli Enterprise Portal toolbar
 - Click the NetView for z/OS Enterprise Management Agent product to see the attribute groups that are applicable for historical data collection

2-31

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Historical data collections require the data warehouse database (named WAREHOUS) to be defined. Typically, for warehouses existing on Windows systems, the following command is issued before the IBM Tivoli Monitoring software is installed:

```
db2 create database warehous using codeset utf-8 territory US
```

Historical data collection (2 of 3)

Tivoli software **IBM**

Historical data collection (2 of 3)

The screenshot shows the 'History Collection Configuration' dialog box. On the left, under 'Monitored Applications', several items are listed: CCC Logs, NetView for z/OS Enterprise Manager, OMEGAMON HE on z/OS, and Tivoli Enterprise Monitoring Server. In the center, there's a table titled 'Select Attribute Group(s)' with columns for 'Group', 'Detailed', 'Summarize Hourly', and 'Summarize Daily'. Below the table are sections for 'Configuration Controls' (Summarization and Pruning) and a 'Pruning' table with columns for 'Pruning' (Yearly, Quarterly, Monthly, Weekly, Daily, Hourly, Detailed data), 'keep', and 'Entries' (Years, Months, Days). At the bottom are 'OK', 'Cancel', 'Apply', 'Help', and 'Advanced' buttons.

- Expand **Monitored Applications**, and click NetView for z/OS Enterprise Management Agent
- Select groups in the table that you want to collect data from
- Select attributes for the groups: collection intervals, locations, summarization, and pruning
- When you have selected all your criteria, click **Start Collection**

2-32

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

To control the quantity of historical data collected, start the IBM Tivoli Monitoring Warehouse Summarization and Pruning Agent as follows:

1. From the system where the Tivoli Enterprise Portal desktop client runs, click **Start > All Programs > IBM Tivoli Monitoring > Manage Tivoli Monitoring Services**.
2. When the list of services is displayed, right-click **Warehouse Summarization and Pruning Agent** and click **Start**.

Historical data collection (3 of 3)

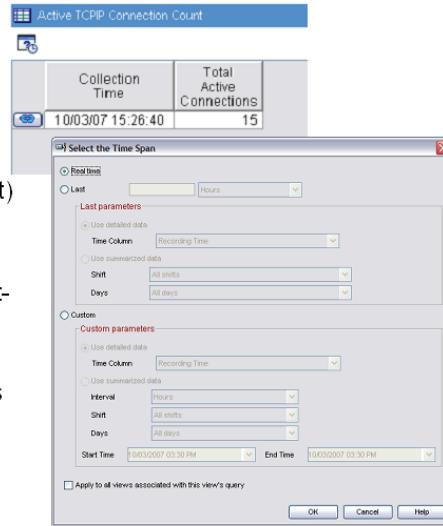
Tivoli software

IBM

Historical data collection (3 of 3)

- After historical data collection is enabled, the upper left corner of qualifying views in Tivoli Enterprise Portal workspaces displays a **Time Span** button 
- Click the icon to extend any existing Tivoli Enterprise Portal view (also called a report) to include historical data
- Tivoli Enterprise Portal reports automatically collects data from both short-term and long-term history, based on the time period that you specify
- You can create summarization data tables (hourly, daily, weekly, quarterly, monthly, and yearly) to reduce the data overload when creating reports
- You can also define pruning intervals to ensure that you save only the needed data

2.33



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

For more information about creating historical reports, see the *IBM Tivoli Monitoring: User's Guide*.

Lesson 6: NetView product integration with IBM Tivoli OMEGAMON XE products

Integration with IBM Tivoli OMEGAMON XE

Tivoli software

IBM

Integration with IBM Tivoli OMEGAMON XE

- The NetView program interoperates with IBM Tivoli OMEGAMON products through the NetView web application and the NetView enterprise agents
- From the NetView web application, you can retrieve performance data for a TCP/IP connection from OMEGAMON XE for Mainframe Networks. You use the Tivoli Enterprise Monitoring Server Web Services interface
- The NetView EMA enables management of both availability and performance data from the Tivoli Enterprise Portal. It uses cross-product links to selected z/OS OMEGAMON XE agents
- The NetView EMA provides linking to OMEGAMON product workspaces
- If the appropriate OMEGAMON products are installed and configured, the links between the NetView EMA workspaces and the OMEGAMON workspaces are operable

2-35

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

A wide variety of NetView data is available through the Tivoli Enterprise Portal. NetView data correlates with information from Tivoli OMEGAMON XE products, including the following items:

- DVIPA
- Sysplex
- Packet traces
- The NetView log
- IP connection data

The NetView program focuses on *availability* management. The OMEGAMON XE products focus on *performance* management.

You can use the Tivoli Enterprise Portal to integrate these solutions for a complete view of network performance and availability data. You can use this information to improve efficiency and to reduce the time needed for problem resolution.

IBM Tivoli OMEGAMON XE cross-product workspace links

IBM Tivoli OMEGAMON XE cross-product workspace links				
NetView Agent Workspace	Target Application or Monitoring Agent	Workspace in Target Application or Monitoring Agent	Attributes Used to Locate Target Workspace	Attributes Used to Filter Data in Target Workspace
DVIPA Connections	IBM Tivoli OMEGAMON XE for Mainframe Networks version 4.1.0 or later	TCP Connections Link	System ID	<ul style="list-style-type: none">• Connection Start Time• Local IP Address• Local Port• Remote IP Address• Remote Port
Session Data	IBM Tivoli OMEGAMON XE for Mainframe Networks version 4.1.0 or later	HPR Connections	System ID	<ul style="list-style-type: none">• Primary Name• Secondary Name
TCPIP Connection Data	IBM Tivoli OMEGAMON XE for Mainframe Networks version 4.1.0 or later	TCPIP Connections Link	System ID	<ul style="list-style-type: none">• Connection Start Time• Local IP Address• Local Port• Remote IP Address• Remote Port
TCPIP Connection Data	IBM Tivoli OMEGAMON XE for CICS® on z/OS version 4.1.0 or later	TCPIP Statistics	System ID	Local Port (converted to an integer in the link expression)
TCPIP Connection Data	IBM Tivoli OMEGAMON XE on z/OS version 4.1.0 or later	System CPU Utilization	Managed system name (Sysplex Name: System ID: "MVSSYS")	None
Telnet Server Configuration and Status	IBM Tivoli OMEGAMON XE for Mainframe Networks version 4.2.0 or later	TN3270 Server Sessions	System ID	Telnet Server Job Name

2-36

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This table shows the OMEGAMON monitoring agents that provide cross-product links. At the time of this publication, the products that provide links are as follows:

- IBM Tivoli OMEGAMON XE for Mainframe Networks 4.1.0 or later
- IBM Tivoli OMEGAMON XE for CICS on z/OS 4.1.0 or later
- IBM Tivoli OMEGAMON XE on z/OS 4.1.0 or later

Lesson 7: Tivoli Enterprise Portal security

User authorization and identification for the Tivoli Enterprise Portal

Tivoli software

IBM

User authorization and identification for the Tivoli Enterprise Portal

- The Tivoli Enterprise Portal logon requires a user ID and password
- Password validation occurs on the operating system that has the Tivoli Enterprise Monitoring Server hub
For z/OS, that is SAF (RACF, ACF2)
- You can set Tivoli Enterprise Portal security for including or excluding Take Action commands

2-38

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can switch from one Tivoli Enterprise Monitoring Server to another. For example, a distributed environment can be defined to use TEMS A, and a mainframe environment defined to use TEMS B. Switching from one Tivoli Enterprise Monitoring Server to another is a Tivoli Enterprise Portal Server reconfiguration process, pointing to another Tivoli Enterprise Monitoring Server.

Password validation occurs on the system with the Tivoli Enterprise Monitoring Server. It is necessary to define the user ID on that system.

Tivoli Enterprise Portal Take Action and system command authorization

Tivoli software

IBM

Tivoli Enterprise Portal Take Action and system command authorization

- Commands route to NetView for running for agents that do not have a command handler
- NetView command authorization occurs as follows:
 1. Table authorization
 2. SAF authorization
 3. Tivoli Enterprise Portal user ID for determining the NetView operator and the command authorization
 4. If command authorization passes, the command runs on the NetView operator
- The messages that are written to the NetView Log provide an audit trail of the commands and the user ID that issued them
- You can use Tivoli Enterprise Portal user reporting to view the NetView Log (3270 or workspace) and Take Action status information

2-39

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

IBM z/OS products that use the Tivoli Management Services components can use NetView for z/OS system command authorization and audit trail support. These products are referred to as agents in these notes.

System commands are routed from the agents to the NetView program. NetView treats these commands as z/OS system commands. NetView prefixes the command that the agent passed with **MVS** before issuing the command. Authorization is performed by using the NetView command authorization table or SAF functions.

Using the NetView OPERSEC SAFCHECK or SAFDEF setting allows more granular z/OS command security. In the *NetView for z/OS Security Reference*, refer to “Chapter 3, Protecting MVS System Commands Using an SAF Product” for more information.

The Tivoli Enterprise Portal user ID is used for determining the NetView operator whose command authorization is checked and where the command runs.

NetView provides the following actions for a system command that is issued by the agent:

- Logging of the command echo and command response to the NetView log
- Logging of the BNH806I audit trail message to the NetView log
- Automation of the command echo and command response

Review your automation and user exits to make sure that the commands and messages that NetView logs are not suppressed.

You can view the NetView Log from the Tivoli Enterprise Portal by logging on to the NetView program from a Terminal View. Alternatively, the NetView for z/OS Tivoli Enterprise Management Agent provides the capability to issue a Take Action command for browsing NetView Log.

Some agents provide their own Take Action commands, known as agent commands. Agent commands have a two-character prefix, such as pp, where pp is the product code. These commands are not sent to the NetView program for command authorization and execution. An example of an agent that provides agent commands is IBM Tivoli OMEGAMON XE for Messaging on z/OS.

Controlling Take Action capability

Tivoli software

IBM

Controlling Take Action capability

At the Administer Users window, you can modify the Take Action permissions for the <Default User> and each defined Tivoli Enterprise Portal operator

The screenshot shows the 'Administer Users' dialog box. In the 'Users' table, there are two entries: 'Default' (User ID: <Default User>, Name: Default, Description: Default, Date Last Modified: 07/30/06 09:57:03, Last Modified By: SYSADMIN) and 'SYSADMIN' (User ID: SYSADMIN, Name: Administration, Description: Administration, Date Last Modified: 06/28/06 14:37:23, Last Modified By: KCJ). The 'Permissions' tab is selected, showing authorities for the 'Default User'. Under 'Authorities', the 'Action' node is expanded, showing 'Agent Management', 'Custom Navigator Views', 'Event', and 'Launch Application'. The 'Feature' node is also expanded, showing 'DE', 'Express', 'History', 'Managed System List', 'Policy', 'Query', and 'Situations'. On the right side of the 'Permissions' tab, 'View' and 'Modify' checkboxes are checked. At the bottom of the dialog are 'OK', 'Cancel', 'Apply', and 'Help' buttons. The footer of the dialog box reads 'IBM Software Group | Tivoli Software © 2011 IBM Corp.'

2-40

Immediate options for Take Action commands are as follows:

- Prevent anyone from issuing Take Action commands from Tivoli Enterprise Portal:
 - This option applies to all users for all products.
 - All commands from OMEGAMON are issued from a 3270 interface.
 - The OMEGAMON 3270 window can be implemented within the Tivoli Enterprise Portal.
- Authorize Take Action command authority to a limited number of trusted users.

Summary

Tivoli software

IBM

Summary

Now that you have completed this unit, you can perform the following tasks:

- Describe the NetView web interface
- Describe the NetView web services gateway interface
- Describe the NetView Management Console (NMC) interface
- Use the Tivoli Enterprise Portal and the NetView EMA workspaces
- Explain how the NetView product integrates with IBM Tivoli OMEGAMON XE products

2-41

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

IBM Tivoli certification and training

In today's global business world, enhancing and maintaining skills is essential to keeping pace with rapidly changing technologies. Businesses need to maximize technology potential and employees need to keep up to date with the latest information. Training and professional certification are two powerful solutions.

Certification

There are many reasons for certification:

- You demonstrate value to your customer through increased overall performance with shorter time cycles to deliver applications.
- Technical certifications assist technical professionals to obtain more visibility to potential customers.
- You differentiate your skills and knowledge from other professionals and stand out as the committed technical professional in today's competitive global world.

Online certification paths are available to guide you through the process for achieving certification in many IBM Tivoli areas. See ibm.com/tivoli/education for more information.

Special offer for having taken this course

Now through 31 December 2011: For having completed this course, you are entitled to a 15% discount on your next examination at any Thomson Prometric testing center worldwide. Use this special promotion code when registering online or by telephone to receive the discount: **15CSWR**. (This offer might be withdrawn. Check with the testing center as described later in this section.)

Role-based certification

All IBM certifications are based on job roles. They focus on a job a person must do with a product, not just the product's features and functions. Tivoli Professional Certification uses the following job roles used:

- IBM Certified Advanced Deployment Professional
- IBM Certified Deployment Professional
- IBM Certified Administrator
- IBM Certified Solution Advisor
- IBM Certified Specialist
- IBM Certified Operator

Training

A broad spectrum of courses, delivery options, and tools helps keep your employees up to date with the latest IBM Tivoli information:

- *Instructor-led training (ILT)*
Live interaction with an IBM instructor, hands-on lab exercises, and networking with your peers from other companies
ibm.com/tivoli/education
- *Instructor-led online (ILO)*
All the benefits of ILT, but savings on travel dollars and training costs
ibm.com/training/ilo
- *Self-paced virtual classes (SPVC)*
Interactive and hands-on exercises on your schedule
ibm.com/training/us/spvc
- *Web-based training (WBT)*
Training anywhere, any time, that saves you money and travel
ibm.com/training/us/tivoli/wbt
- *Multimedia library*
Modules supporting new and experienced learners with fully animated multimedia clips, step-by-step audio, and companion text
ibm.com/software/tivoli/education/multimedialibrary
- *IBM Education Assistant*
More specific, granular web-based training with individual presentations on specific topics
www-01.ibm.com/software/info/education/assistant/
- *Corporate Education Licensing Program (CELP)*
Solutions for large IBM customers who need to adopt IBM Tivoli's tools and technologies
ibm.com/training/us/tivoli/celp
- *Tivoli training paths*
Course maps with flow charts and course descriptions to help you find the right course
ibm.com/training/us/tivoli/path



IBM Tivoli NetView for z/OS 6.1: Automation Techniques

Student's Training Guide

Course: TZ213 ERC: 1.0

August 2011

© Copyright IBM Corp. 2011. All Rights Reserved.

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

IBM, the IBM logo, and ibm.com are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in this publication to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth, savings or other results.

Printed in Ireland

Table of contents

Unit 1: Introduction to NetView automation

Introduction	1-2
Objectives	1-2
Lesson 1: Overview of automation concepts	1-3
Classes of automation	1-4
SMF type 30 record automation	1-4
System automation	1-5
Network automation	1-6
Types of events	1-7
Single-system automation tasks	1-9
Single-system versus multisystem automation	1-11
Multisystem automation	1-12
NetView sysplex and DVIPA	1-13
NetView-to-NetView connections	1-15
Basics of automation	1-16
Resource monitoring	1-18
VTAM interfaces	1-19
TCP/IP interfaces	1-20
Policy-based automation	1-22
NetView and z/OS communications	1-24
Open system support	1-25
Lesson 2: Events and tasks	1-27
Events and NetView tasks	1-28
Primary POI Task (PPT)	1-29
Operator Station Task (OST)	1-31
Automated Operator Task (AOST)	1-32
RMTCMD sessions	1-33
Virtual Operator Task (VOST)	1-35
LIST STATUS=TASKS	1-36
Terminal Access Facility	1-37
Other major tasks	1-38
NetView data interfaces	1-39
Summary	1-41

Unit 2: Message automation topics

Introduction	2-2
Objectives	2-2
Lesson 1: NetView and z/OS messages	2-3
Message flow: z/OS to NetView	2-5
NetView SSI message routing	2-7
Command flow between z/OS and NetView	2-8
Command flow with extended MCS consoles	2-9
NetView program-to-program interface	2-10
Automation in a sysplex environment	2-11
Extended MCS consoles in a sysplex	2-12
Console management commands	2-13
EMCS example	2-14

MCS and message dispersal2-15
Descriptor codes for z/OS messages2-16
Route codes for z/OS messages2-17
Message processing facility (MPF) for z/OS2-18
MPF lists for z/OS: MPFLSTxx2-19
MPF implementation and use2-20
Overview of MPFLST statements2-21
MPF .DEFAULT example2-22
MPF list example2-23
Lesson 2: Extended message management2-24
Automation flow with message revision2-25
REVISE command2-27
Message revision features2-28
Extended message management2-29
Trapping messages in the MRT2-30
MRT search order2-32
MRT search order example2-33
MRT actions2-34
Using the MRT to modify messages2-35
Revision examples2-36
Modifying messages with the MRT2-37
System console example2-38
Revision variables2-39
Student exercise2-40
Lesson 3: Message assignments2-41
Message assignments (2 of 2)2-42
Types of messages2-43
Primary receiver2-45
Operator groups2-46
ASSIGN and LIST commands2-47
ASSIGN MSG examples2-48
ASSIGN GROUP examples2-50
Displaying the primary receiver2-51
Displaying ASSIGN GROUP definitions2-52
Student exercise2-53
Summary2-54

Unit 3: NetView commands for facilitating automation

Introduction3-2
Objectives3-2
Lesson 1: NetView-to-NetView communication3-3
RMTCMD and TAF3-4
RMTCMD session for an operator3-5
Shared RMTCMD session3-6
CNMSTYLE definitions3-7
RMTSYN definition examples3-8
Query RMTCMD sessions: Outbound3-10
Query RMTCMD sessions: Inbound3-11
Lesson 2: Routing commands3-12
Example of routing commands3-13
Student exercise3-14
Lesson 3: Timer commands3-15
Scheduling a timer3-16
Supported commands3-17

Routing commands to the PPT	3-18
Timers for active monitoring	3-19
AT examples	3-20
EVERY examples	3-21
AFTER example	3-22
ROUTE and TIMEFMSG operands	3-23
EVERYCON operand	3-24
CHRON example	3-25
Saving and restoring timers	3-26
Miscellaneous timer topics	3-27
Displaying timers	3-28
LIST TIMER example	3-29
TIMER command	3-30
TIMER display example	3-31
Creating an AT timer	3-32
Displaying remote NetView timers	3-33
Displaying the list of remote domains	3-34
Filtering timer data	3-35
Purging timers	3-36
Student exercise	3-37
Lesson 4: Command Revision Table (CRT)	3-38
Student exercise	3-40
Summary	3-41

Unit 4: Managing the NetView automation table

Introduction	4-2
Objectives	4-2
Lesson 1: Automation table overview	4-3
Automation table overview (2 of 2)	4-4
Automation table basic approach	4-5
Segmented automation tables	4-6
Multiple automation tables	4-7
Lesson 2: Automation table management	4-8
Loading an automation table	4-9
AUTOTBL examples	4-11
AUTOTBL swap and insert	4-12
Automation table load notes	4-13
Loading an automation table from CNMSTYLE	4-14
AUTOTBL STATUS example	4-15
Automation table management tool	4-16
AUTOMAN main panel	4-17
AUTOMAN table insert	4-18
Testing an automation table	4-19
Displaying automation table statistics	4-20
AUTOCNT example	4-21
AUTOCNT REPORT=MSG,STATS=SUMMARY example	4-22
Student exercise	4-23
Summary	4-24

Unit 5: Coding an automation table

Introduction	5-2
Objectives	5-2

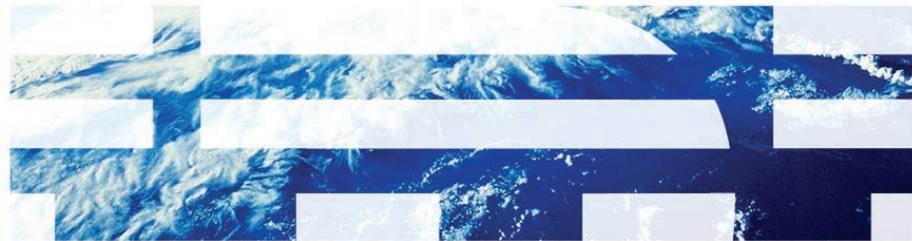
Lesson 1: Coding automation table statements 5-3
Characteristics of events 5-4
Automation action choices 5-5
Routing automation actions 5-6
Processing the event 5-8
Automation table statements 5-9
IF-THEN overview 5-11
IF-THEN condition 5-12
Examples of IF-THEN condition items 5-13
Examples of IF-THEN actions 5-14
IF-THEN EXEC action 5-15
IF-THEN ROUTE parameters 5-16
IF-THEN ROUTE examples 5-17
BEGIN-END block 5-18
ALWAYS statement 5-19
IF-THEN example 1 5-20
IF-THEN example 2 5-21
IF-THEN example 3 5-22
IF-THEN example 4 5-23
IF-THEN example 5 5-25
IF-THEN example 6 5-26
IF-THEN example 7 5-28
IF-THEN example 8 5-29
IF-THEN example 9 5-30
IF-THEN example 10 5-31
IF-THEN example 11 5-32
Automation of multiline messages 5-33
BEGIN-END example 5-34
Testing for unsolicited messages 5-35
Testing for commands 5-36
SYN statement 5-37
Conditionally including statements 5-39
CNM493I message 5-41
Controlling CNM493I 5-43
Student exercise 5-44
Lesson 2: Management Services Unit (MSU) 5-45
MSU types 5-46
MSU structure 5-47
Major vectors 5-48
NetView automation of alert data 5-49
MSUSEG example 1 5-50
MSUSEG example 2 5-51
SNMP trap automation task 5-52
Summary 5-53

Unit 1: Introduction to NetView automation

IBM

Unit 1:

Introduction to NetView automation



© 2011 IBM Corp.

Introduction

This unit introduces you to many of the automation concepts, such as monitoring of resources, automation of system and network resources, single-system and multisystem automation, and so on.

Objectives



Objectives

When you complete this unit, you can perform the following tasks:

- Define passive versus active monitoring
- Define system versus network automation
- Describe single-system and multisystem automation
- Identify event types that can be processed and how they arrive in NetView.

Lesson 1: Overview of automation concepts



Lesson 1: Overview of automation concepts

- System automation
- Network automation
- Performance automation
- Types of events
- Single-system automation
- Multisystem automation
- Active monitoring
- Passive monitoring
- Policy-based automation
- Non-SNA automation
- Non-NetView automation

1-3

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This lesson introduces several automation topics as follows:

- System versus network automation
- Single-system versus multisystem automation
- Active monitoring versus passive monitoring
- And more

Classes of automation

IBM

Classes of automation

- System automation
 - Applications
 - Jobs
 - And so on
- Network automation
 - Physical devices
 - Network connectivity
 - And so on
- Performance automation
 - OMEGAMON products

1-4

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

System automation focuses on system resources, such as applications, batch jobs, started tasks, and so on. IBM® Tivoli® System Automation for z/OS® (SA z/OS) is an IBM product that you can purchase for automating your systems and sysplexes (system complexes).

SMF type 30 record automation

The MVS IEFACRT SMF installation exit receives control from the system when a job or job step ends, either normally or abnormally. The NetView® program provides a sample IEFACRT exit (CNMSMF3E) that passes data across the program-to-program Interface (PPI) to a receiver. The receiver issues a message containing the data that can be processed by using NetView automation facilities. This enables quicker and more consistent responses for jobs that end abnormally.

Network automation focuses on network resources, such as System Network Architecture (SNA) lines, physical units (PUs), connectivity between systems, and so on. The Automated Operations Network (AON) component of IBM Tivoli NetView for z/OS provides automation for your VTAM and TCP/IP networks. AON is available with NetView.

Performance automation focuses on how well the resources handle their current workload. For example, with SA z/OS and OMEGAMON®, you can monitor for exception conditions and take additional actions. You can start another instance of an application to improve throughput.

System automation



System automation

- IBM Tivoli System Automation for z/OS (SA z/OS)
- Monitoring, starting, stopping, and failure recovery of systems and applications:
 - DB2®
 - IMS™
 - CICS®
 - TWS
 - OMEGAMON®
 - WebSphere®
 - And more
- Capability of automating Initial Program Load (IPL) or shutdown of systems
- And more

1-5

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

SA z/OS uses policy to automate resources in a system and sysplex (system complex). SA z/OS handles the starting, stopping, monitoring, and failure recovery of the resources for DB2®, IMS™, and so on, across all systems in your sysplex or subplex. Systems are also handled like a resource and can be started (IPL) and stopped (shutdown).

SA z/OS provides you with default policy that includes relationships. For example, you can issue a start request against an application, and SA z/OS starts other prerequisites or dependent applications.

The SA z/OS automation policy is defined with the customization dialog, a set of Interactive System Productivity Facility (ISPF) panels.

Network automation



Network automation

- Automated Operations Network (AON)
Component of NetView
- Monitoring, starting, stopping, and recovery of network resources:
 - VTAM subarea
 - SNBU
 - X.25
 - TCP/IP

1-6

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The AON component of NetView provides default automation for your network resources. AON monitors, starts, stops, and recovers your VTAM SNA and TCP/IP resources, for example.

Types of events

Types of events

- Messages
- SNA MSUs
 - Alerts: Problem notification
 - Resolutions: Problem-fixed notification
- TCP/IP traps
 - Converted to MSUs
- Non-SNA agent data
 - Converted to messages or MSUs
- Common Base Events (CBEs)
 - Converted to messages
- NetView commands
- All events that are converted to messages or MSUs

The diagram illustrates the flow of events into the NetView automation system. At the top right is the IBM logo. Below it is a pink cloud containing a blue rectangle labeled 'NetView' with 'automation table' written inside. Arrows point from various sources to this central cloud:

- A z/OS icon points to the cloud with an arrow labeled 'messages'.
- A person at a computer icon points to the cloud with a dashed arrow labeled 'CBEs'.
- A person with a wrench icon points to the cloud with a dashed arrow labeled 'TCP/IP'.
- A blue cloud labeled 'SNA' points to the cloud with an arrow labeled 'MSUs'.
- A blue cloud labeled 'CEI' points to the cloud with an arrow labeled 'CBEs'.
- A blue cloud labeled 'TCP/IP' points to the cloud with an arrow labeled 'SNMP traps'.

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

1.7

Each type of event is translated into a message or a Management Services Unit (MSU) that NetView automation is to process. The most common form of event is a message. Most z/OS products issue messages that pertain to events within the product, system, or network.

MSUs are SNA events that are represented as alerts (problem) or resolutions (problem fixed). MSUs use an architected structure that is similar to an envelope for data. TCP/IP traps, which are converted to MSUs, can be generated by agents and sent to NetView for automation. Non-SNA data is packaged in MSUs or as messages by an agent and sent to NetView for automation.

Common Base Events (CBEs) are a standardized representation of events across hardware and software. CBEs are converted to messages before being sent to NetView automation.

Commands are also eligible for automation. All NetView commands drive the NetView automation table. Most customers add a statement to the automation table to ignore command automation. This discussion occurs later in the class. NetView also provides a *command revision exit* that allows you to modify the text of a z/OS command before it is issued.

Events that arrive at NetView are tested against statements in the automation table. All events are translated to either a message or an MSU before being run through the automation table. If a match is found in the automation table, the event can be displayed or suppressed, and one or more actions can be taken.

This class shows you the following skills:

- How to trap events in the automation table
- Controlling the processing of the event (for example, display or suppress it)
- How to take actions
- Where to route the actions

Single-system automation tasks



Single-system automation tasks

- Suppress messages and block alerts
- Consolidate consoles
- Consolidate commands
- Schedule commands
- Automatically respond to events
- Establish coordinated automation
- Improve operator interfaces
- Run screen operator commands



Use NetView automation to increase speed and accuracy of operators who process information



Reduce operator workload by using NetView to perform some management tasks



Adapt operator interface to new environment and reduced workload

1-8

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide illustrates the steps in automation of a single z/OS system. The beginning stages involve reducing the amount of work your operations staff is asked to perform. Later stages involve automation to recover failing resources, further reducing the workload on your operations staff.

Many products send their messages to the system console. Most of those messages are informational. The first step in automation is determining the messages that are important and suppressing those that are not. By using automation to reduce the number of messages displayed on the console, the operations staff can be more effective.

The second step in automation is console consolidation. By reducing the number of messages displayed on the consoles in the first step, you reduce the number of consoles, further improving the effectiveness of your operations staff.

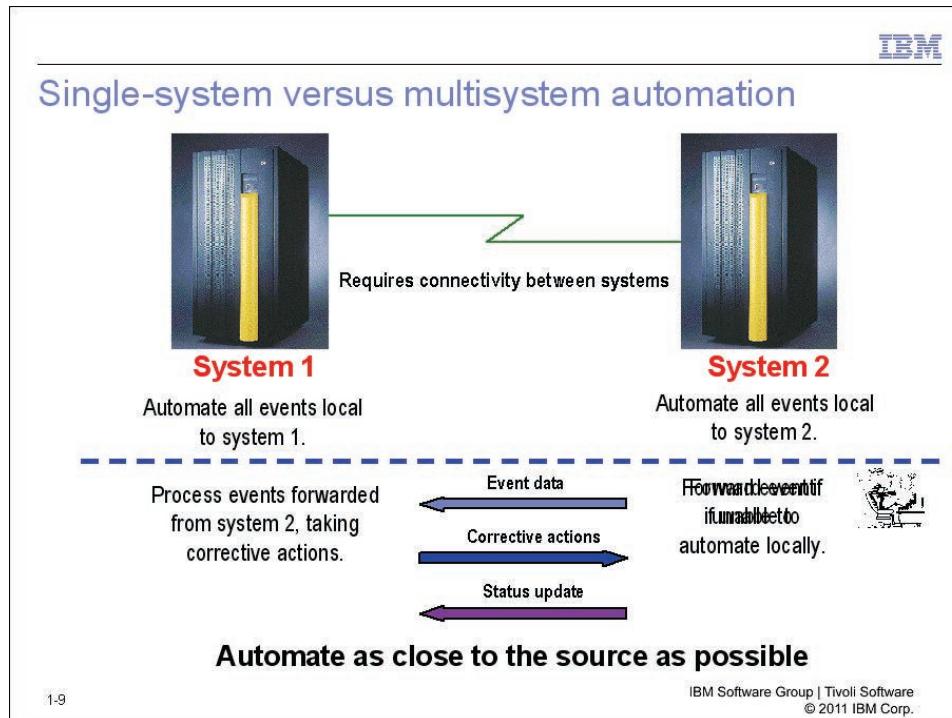
The third step in automation is consolidating commands into REXX EXECs. Suppose you have an application failure and your operations staff must issue the same sequence of commands each time. You can consolidate the sequence of commands into a REXX routine and have the operations staff run the REXX routine instead. Afterward, you can use the REXX EXECs and automate the recovery of the application when it fails instead of relying on the operations staff to manually perform the recovery.

You can create additional REXX EXECs to monitor critical resources by scheduling the EXECs with NetView timers. If resources fail, you can also automate a logical grouping that could include system and network resources.

Suppose an application fails, and you need to schedule a database archival to a workstation. You can coordinate the tasks required between the system, network, and distributed (workstation) resources to automate the recovery of the application as a logical entity. You can also coordinate the application recovery with an email notification to your shift supervisor.

NetView provides the infrastructure to accomplish all of your automation and also automation of your network resources. With SA z/OS, you can also automate your system and sysplex resources.

Single-system versus multisystem automation



After you complete automation of a single system, you can move to the next phase, which is multisystem automation. In multisystem automation, you use the single-system automation in each of the systems to automate the events for each system. This strategy is called automating as close to the source (of the event) as possible.

In multisystem automation, you designate one of the systems to be the focal point system. Each single system attempts to automate the events it receives. Some of the events might require that they be forwarded to the focal point for automation. The focal point processes the event and attempts to automate by sending commands to the remote system.

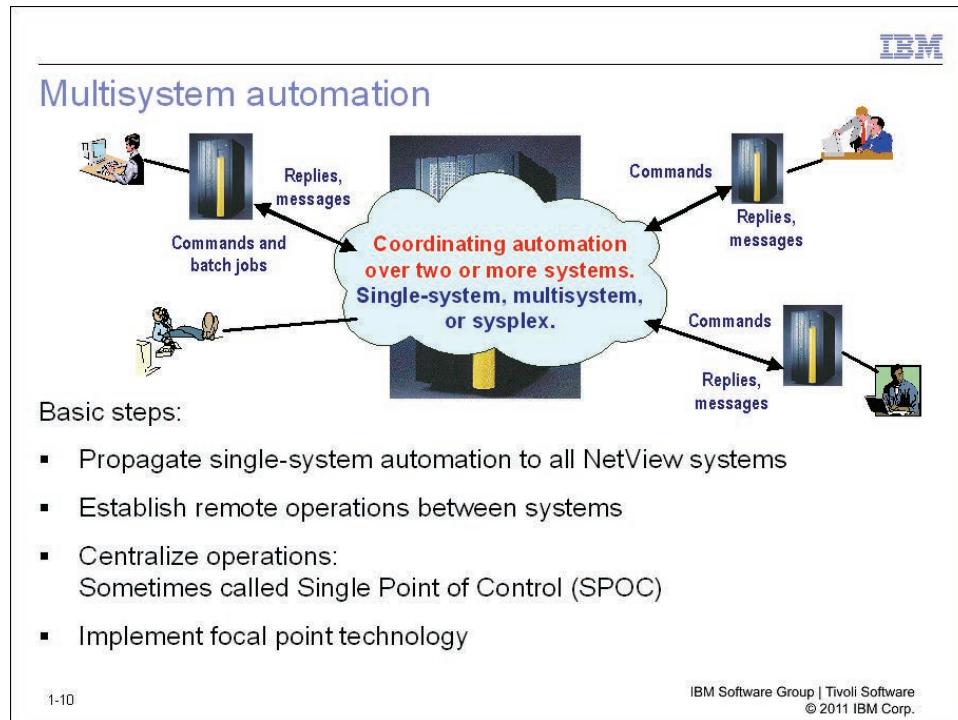
For multisystem automation, each of the systems must be connected to the focal point. For example, the connection could be a traditional VTAM CDRM, a TCP/IP connection, or an XCF connection if the systems are connected in a sysplex.

NetView-to-NetView connectivity must also be defined.



Note: Multisystem automation is not the same as sysplex automation. Sysplex automation occurs when you exploit the capabilities that a sysplex provides.

Multisystem automation



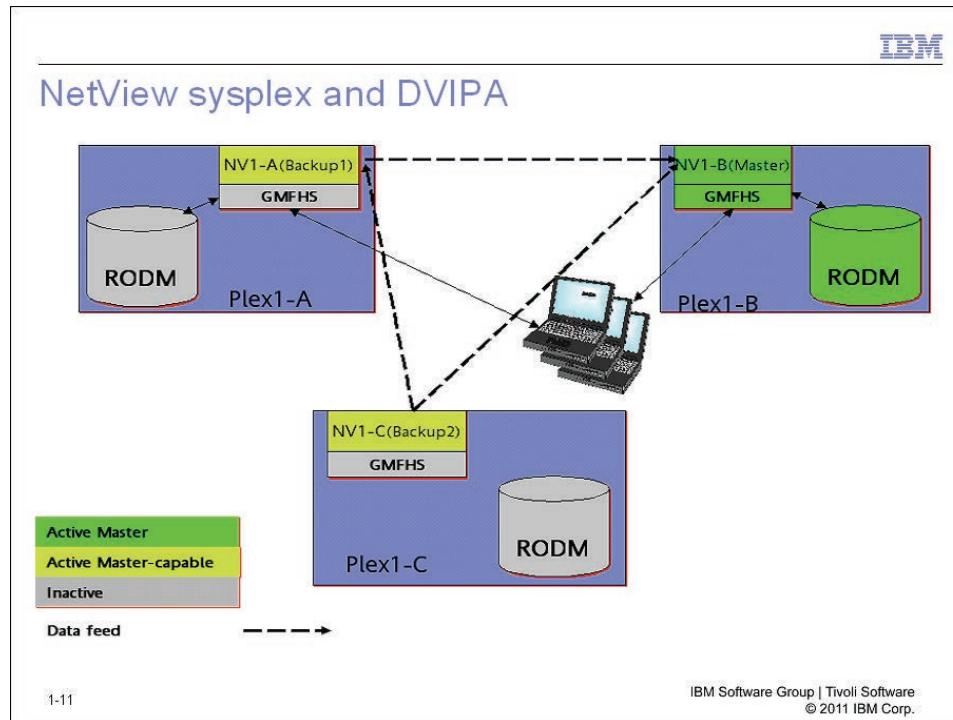
Multisystem automation, whether a single site or multiple sites, has a few additional steps to those discussed for single-system automation.

Multisystem automation involves establishing coordinated automation between two or more z/OS systems. The coordination enables the operation of resources that cannot be automated locally on a single system.

In many cases, the focus is to centralize operations, even when dealing with remote (distributed) host systems. In the case of no staff at the remote location, data pertaining to the remote systems is forwarded to the central site (focal point). The data could include problem reports that cannot be automated at the remote site.

In other cases, facilities with automation on one system can inquire about status of another system, forward information about other systems, initiate actions on other systems, and check the results of such actions. Multiple systems can be in different operational sites and function within different sysplexes.

NetView sysplex and DVIPA



The NetView program provides high availability sysplex management for complex system interactions. You can use it to manage and display information about your sysplex.

Automatic failover to another NetView program that can monitor the sysplex in case of outage is also provided. NetView can also monitor sysplex and system resources as follows:

- Sysplexes
- Coupling facilities
- z/OS images
- TCP/IP stacks, IP interfaces
- Dynamic virtual IP addresses (DVIPAs)
- Telnet servers and ports
- Central processor complexes
- Logical partitions
- Open Systems Adapter (OSA)
- HiperSockets™ adapters

A master NetView program can provide management for systems outside of the sysplex as well as for another sysplex. This NetView program is known as an enterprise NetView program. Additional configuration is necessary for the enterprise NetView program to manage systems that are outside of the sysplex. DVIPA information is restricted to sysplex management.

A dynamic virtual IP address (DVIPA) can be defined for the master NetView program. This address is associated with the master NetView program, so that connection requests to this address always go to the master program. This address can be used by applications for contacting the master NetView program regardless of the system that it runs on within the sysplex.

XCF services can automatically provide information about other NetView programs within the sysplex. If an outage occurs, the NetView program is able to failover to another NetView program in case of an outage. By using XCF services, a master NetView program collects and processes data from other NetView programs within the sysplex to provide a single point of control.

NetView-to-NetView connections

NetView-to-NetView connections

- XCF Services within a sysplex
- RMTCMD session:
 - Platform for message automation
 - Started by operators or automated operators
 - Most common connection
 - SNA LU 6.2 or TCP/IP
- TAF full screen 3270:
 - Functionally stabilized, but still in use
- TN3270 connection
- LUC for NPDA alert and NLDM session data
- MS Transport for NPDA alert and Operations Management data:
 - Platform for alert automation



1-12

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

By default, the NetView program participates in an XCF group that enables communication with other NetView programs in your sysplex environment for supporting discovery manager

Multisystem automation requires connectivity between your z/OS systems and your NetView domains. If you have NetView domains outside your sysplex, you can use RMTCMD to define the connections. The sessions can be over SNA LU6.2 or TCP/IP. In RMTCMD sessions, and you can connect to a target domain and issue commands.

With the terminal access facility (TAF) component of NetView, you can connect to other z/OS applications, such as CICS, IMS, and NetView to receive messages and issue commands. TAF supports both line-mode and full-screen mode sessions.

Other types of connections that NetView uses include connections as follows:

- LUC sessions for NPDA and NLDM data
- MS Transport for alerts (MSUs) and Operations Management data



Note: SA z/OS uses XCF within the sysplex for communication between Automation Manager and Automation Agents. XCF is also used for communication between SA z/OS Automation Agents within the same XCF group. RMTCMD will be used between Automation Agents that are not members of the same XCF group.

Basics of automation

IBM

Basics of automation

- Resource status
 - Current
 - Required or desired
 - Typical automation comparison of current versus desired status
 - Take action, if needed
- Actions, which can be one or more as follows:
 - Ignore, and take no action
 - Monitor
 - Start or recover a failed resource
 - Stop
 - Notify appropriate personnel

1-13

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide shows the basic elements of any automation. Events that drive automation routines typically indicate the current status of the resource. In most cases, the automation routines also detect the desired status that defaults to *active*. In some cases, the desired status is for the resource to be down. The norm is for automation to attempt to activate or recover a resource.

Basic automation actions include:

- Take no action by ignoring the event.

The event might be a result of a planned outage. Or the event might be a result of another outage already being automated.

- Monitor the resource.

In some cases, the resource might recover by itself. If that occurs, do not use automation to attempt recovery. In other cases, you can monitor critical resources to prevent missed outages.

- Start (activate) the resource.
- Stop (deactivate) the resource.
- Recover from a failure.
- Notify appropriate personnel. For example, send an email to another group in your shop or to a vendor. In the notification, inform them of an outage or the actions that automation took to recover from a failure.

NetView provides an INFORM policy for notifications. IBM Tivoli System Automation for Integrated Operations Management (SA IOM) also provides policy and functions for notifications.

Most automation products also provide additional features:

- Perform a *root cause analysis*, identifying if the current event is a symptom of another event.
- Perform *thresholding* to prevent automation from attempting resource recovery for a resource that fails continually. In that case, you have a problem that requires **operator intervention** before automation should be allowed to continue.
- Support *service periods* where you can define time intervals (for example, third shift) where it is acceptable for the resource to be down.

Resource monitoring

IBM

Resource monitoring

- Passive monitoring: Waits for events to update resource status
 - Also called reactive monitoring
 - Resource status part of an unsolicited event
 - Events trapped in the NetView automation table
- Active monitoring: Polls current resource status on a timed basis
 - Also called proactive monitoring
 - Resource status solicitation
 - Timer commands: AT, EVERY, AFTER, CHRON
- Another approach: A combination of both types of monitoring



1-14

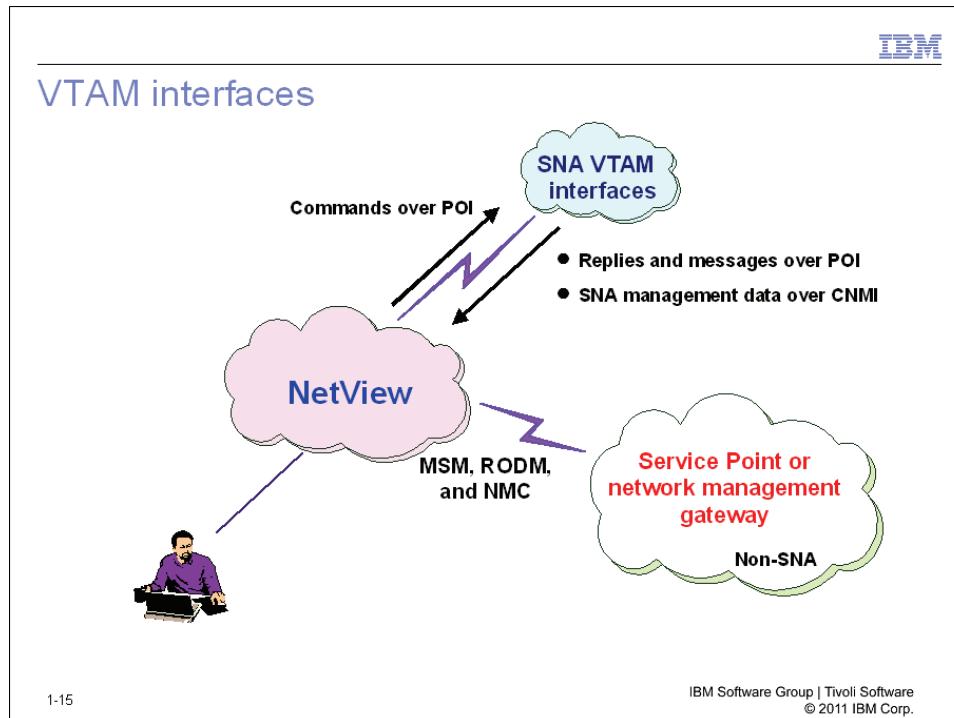
IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Up to now, this class has only discussed waiting for events to occur to drive automation. This is known as **passive monitoring**. This is also called *reactive monitoring* because the automation reacts to events as they are processed by the automation table. The events that drive passive monitoring are *unsolicited*. That is, they are the result of a resource failure and not related to any operator command or request.

You can monitor resource status by scheduling NetView timers to check on the status of one or more resources. This policy is known as **active monitoring**. It is also called *proactive monitoring* because the automation routines proactively check resource status instead of waiting for events. Active monitoring generates *solicited* events. That is, they are the result of a command that is issued to check the status of a resource.

Automation that performs both active and passive monitoring provides coverage because of polling for status and also reacting to events.

VTAM interfaces



NetView has access to SNA and non-SNA data. It communicates with SNA resources using two following VTAM interfaces:

- Programmed Operator Interface (POI) for messages and commands
- Communications Network Management Interface (CNMI) for SNA MSUs



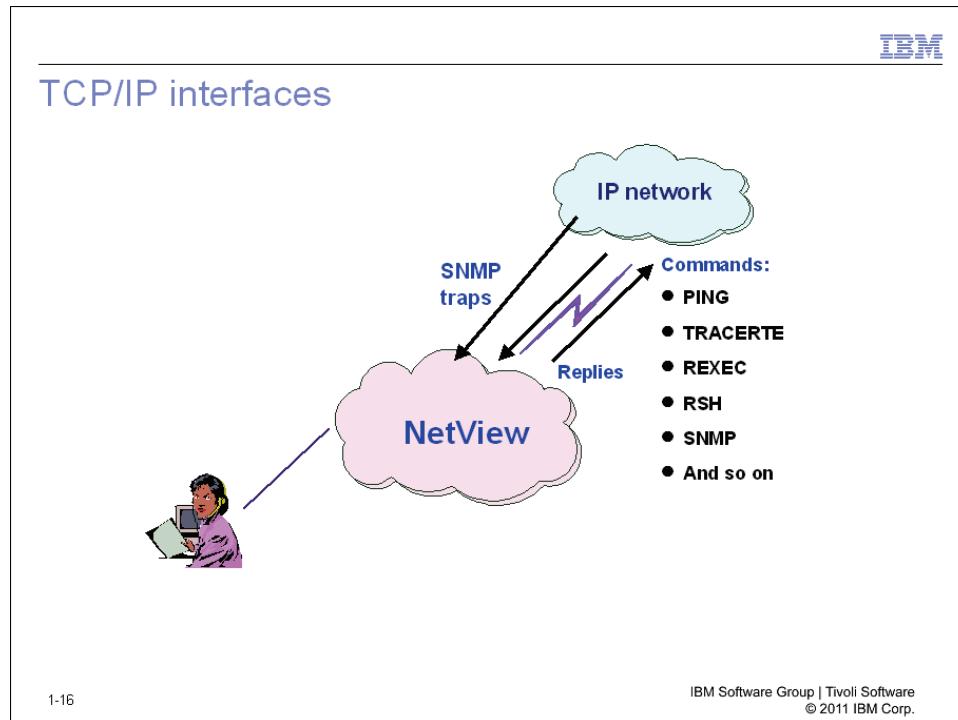
Note: If the NetView-VTAM POI connection is not active, z/OS uses MPF to handle VTAM messages. Discussion about MPF occurs later.

NetView traps both SNA data records, called Management Services Units (MSUs), and VTAM messages. It analyzes them and takes appropriate actions. NetView can automate actions pertaining to any SNA network resource.

For non-SNA networks, an IBM Service Point (called a network management gateway) provides access to the non-SNA resource data. The gateway converts the data to SNA MSUs, which are then transported to NetView, the focal point.

You use the MultiSystem Manager (MSM) component of NetView to obtain data from non-SNA environments. MSM acts as manager to the non-SNA networks that IBM Tivoli Network Manager or user-written Open Topology agent manage. Some agents can inform MSM of resource status changes. Others require MSM to poll for resource status.

TCP/IP interfaces



NetView also has access to TCP/IP network resource data. The Event/Automation Service (E/AS) is an optional address space that runs in the same z/OS system as NetView and the TCP/IP address space. Traps generated by TCP/IP devices can be automated in NetView as follows:

- E/AS can translate SNMP traps from the IP network to alerts and forward them to the NetView automation table.
- SNMP traps can arrive directly in the automation table without E/AS.

Proactive monitoring with the AON/TCP component of NetView is also possible.

You can take following actions against TCP/IP resources:

- Use TN3270 to log in to a remote system.
- Issue any Unix System Services command.
- Issue a PING or TRACERTE command to test the connectivity to the node.
- Issue an SNMP command to retrieve or set MIB values, which can assist with management of SNMP-supported resources.
- Issue a REXEC or RSH command to run UNIX commands at an IP node that uses differing levels of security.
- Issue the RMTCMD command to send system, subsystem, and network commands to a remote NetView host for processing.
- Issue an IPCMD command to issue a TCP/IP command directly to the local TCP/IP stack.
- Issue an IPLOG command to log a message at the IP node.
- Issue a TCPCONN command to display detailed TCP connection data.
- Issue a PKTS command to retrieve TCP/IP packet trace data for debugging purposes.
- Issue the SOCKET command to retrieve information about TCP/IP stacks or to run stack services. You can use it in TCP/IP applications that are based on the NetView program.

Policy-based automation



Policy-based automation

Automation policy definitions are as follows:

- Eligibility for automation
- Proactive monitoring
- Thresholds
 - Prevention of wasting CPU cycles
- Resource relationships
- Service period windows
 - For example, not recovering a resource if it fails during off-shift hours
- Notification methods
 - For example, NetView INFORM policy for beeper and email support
- And so on

1-17

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

With policy-based automation, you can define parameters that pertain to the automation separately from the automation itself. When an event occurs, the automation routines consult the policy. The following determinations might occur:

- Automation
- Thresholds
- Notification recipients about the failure
- Method of notification
- Possible additional conditions

Notifications can use NetView INFORM policy, SA IOM, or other products.

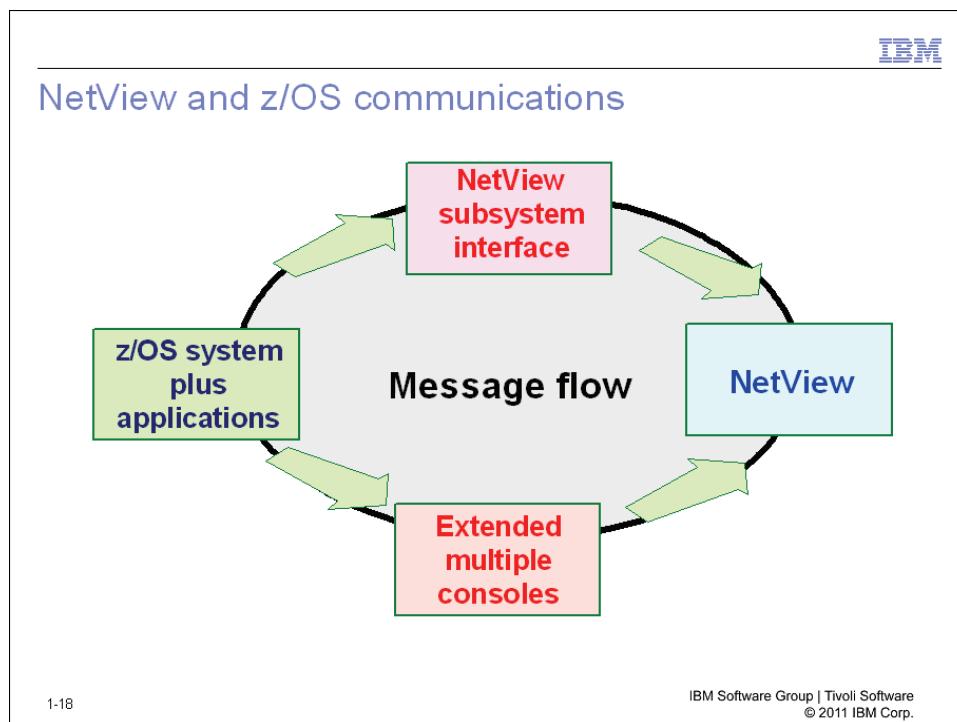
Automation concepts apply to both system and network resources. Automation policy definitions and implementation can differ. For example, SA z/OS automation policy is defined using ISPF-based customization dialogs whereas AON automation policy is defined using flat files located in DSIPARM.



Note: Several NetView components use policy files located in DSIPARM and controlled by **POLICY**. statements in CNMSTYLE. For example, the sysplex topology function detects

TCP/IP stacks and updates stack policy definitions. The stack policy is then used by components, such as the Tivoli Enterprise Portal, web applications, automation, and so on.

NetView and z/OS communications



NetView is an application that runs directly in its own address space under the z/OS operating system. NetView is defined as a z/OS subsystem.

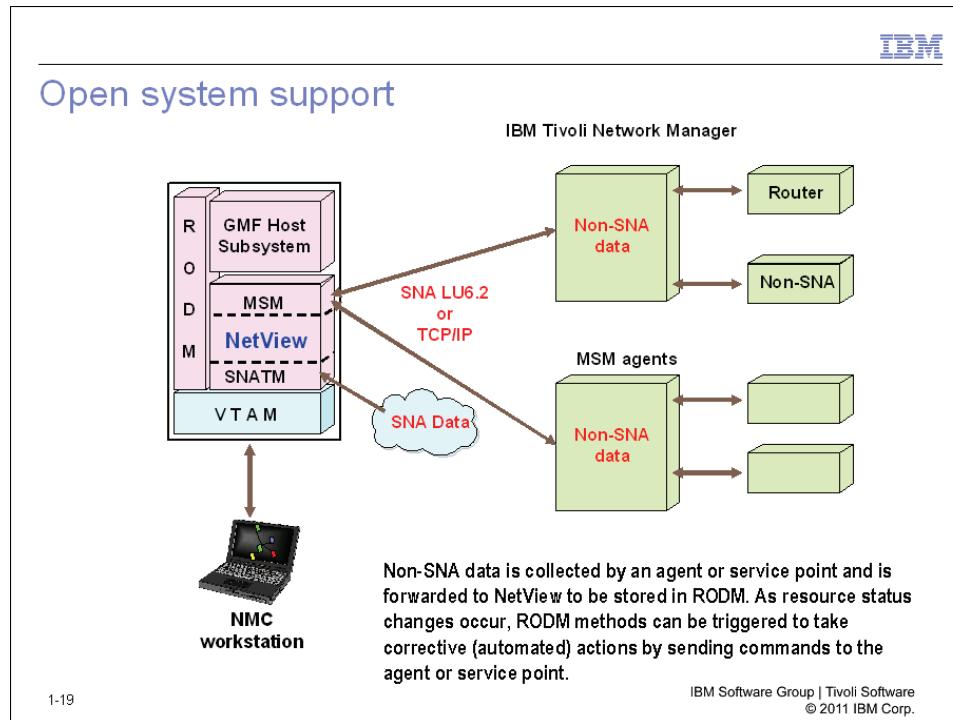
Communication with the z/OS operating system requires a second address space that buffers data to and from z/OS. This address space, called the NetView subsystem interface (SSI), is the direct interface to the z/OS subsystem interface. You can use the SSI to transfer commands and messages between z/OS and the NetView application.

As shipped, the NetView program uses extended multiple console support (EMCS) consoles for tasks that must issue MVS system operator commands. EMCS consoles are dynamically defined and not restricted to a maximum of 99 consoles.

EMCS consoles are used for sending commands from the NetView program to the MVS operating system and receiving messages from MVS. The console name must be unique for each MVS or each sysplex. To avoid console name conflicts, you can use the NetView program to assign a permanent, unique console for each operator by the ConsMask statement in CNMSTYLE.

These interfaces permit NetView access to messages that arrive from the z/OS system tasks and also the z/OS applications.

Open system support



Open systems support requires that any type of network can be managed and automated. NetView can receive and process data that are needed for management and automation of *open systems*. It uses components as follows:

- Resource Object Data Manager (RODM)

RODM is a high-speed data cache that stores the current status of any resource within the complex in data spaces. Agents, such as those that MSM supplies, collect the necessary data and forward it to NetView to be stored in RODM. RODM methods can be triggered to act upon this data.

- MultiSystem Manager (MSM)

MSM enables NetView to act as a manager to deal with IBM Tivoli Network Manager and user-written Open Network Topology agent in the enterprise. Data that MSG retrieves is stored in RODM.

- NetView Management Console (NMC)

NMC is a graphical interface that displays configuration (topology) and status information that RODM stores on a Java-based workstation.

- Graphic Monitor Facility Host Subsystem (GMFHS)

GMFHS provides communication between NMC and RODM.

- SNA Topology Manager (SNATM)

SNATM collects SNA subarea and APPN data and stores it in RODM.

NetView can also forward data to other Tivoli products, such as Tivoli Netcool®/OMNIbus program, IBM Tivoli Enterprise Console, and IBM Tivoli Business Service Manager.

An agent is an application running on a network device that communicates with NetView over SNA LU 6.2 or TCP/IP at the mainframe host. The agent can send data (for example, status or topology data) to NetView and receive commands from NetView.

When RODM receives updates concerning the status of a resource, NetView can act on that information and automate responses. Two ways of triggering NetView automation are as follows:

- An alert created for processing by the automation table
- A RODM method (program) invoked automatically

Lesson 2: Events and tasks

IBM

Lesson 2: Events and tasks

Types of events

Messages	SNA MSUs (alerts)	Non-SNA agent data	TCP/IP traps	Common Base Events (CBEs)

A blue bracket groups the first four event types under the heading "NetView tasks involved". Below this bracket is a screenshot of the NetView automation table interface, showing various event entries with status indicators like red dots and arrows.

NetView tasks involved

1-20

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This lesson discusses the types of events that NetView can process and the tasks that are required. Each event translates into a message or an alert for processing by the automation table.

Events and NetView tasks



Events and NetView tasks

Events can arrive at several tasks:

- Subsystem interface (SSI)
CNMCSSIR routes these events
- Program-to-program interface (PPI)
- Primary POI task (PPT)
- TAF sessions (FLSCN and OPCTL)
- Operator station tasks (OST)
- Automated operators (AOST)
- Virtual operators (VOST)
- RMTCMD sessions (distributed AOST)
- Communication Network Management Interface (CNMI)
DSICRTR routes these events
- Hardware Monitor (NPDA) main task (BNJDSERV),
Over LUC or MS Transport tasks

1-21

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Events can arrive in NetView from several sources as this slide lists. Examples are as follows:

- z/OS messages coming across the SSI interface
- Applications sending and receiving data using queues within the PPI
- Unsolicited VTAM messages coming in through the PPT task
- Messages from remote systems arriving through RMTCMD sessions

This list is a small subset of the NetView tasks. This class discusses NetView tasks that pertain to automation. Issue a **LIST STATUS=TASKS** command to display all tasks that are defined to NetView.

Primary POI Task (PPT)

Primary POI Task (PPT)

- Only one PPT per NetView
 - Defined to VTAM as PPO
 - If running multiple NetView instances, all others to be SPO
- Unsolicited VTAM and NetView messages routed to PPT
- Control of scheduling of timers
- Capability to receive messages and run commands
 - Restrictions on commands
 - For example, any REXX EXEC that issues WAIT to be cancelled
- If stopped, NetView to end abnormally (ABEND)

Minimize use of PPT when possible. Assign messages to an authorized receiver for processing.

```
graph TD; MNT[MNT] --> OST[OST]; MNT --> AOST[AOST]; MNT --> PPT[PPT]; PPT --> VTAM[VTAM]; VTAM --> POI[POI]
```

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The Primary Programmed Operator Interface (POI) Task, otherwise known as PPT, processes unsolicited VTAM and a subset of NetView messages. If the system administrator who creates your automation table statements does not supply message routing logic, the PPT designates the recipient operator. If no operator is found, the message routes to the system console.

Each NetView has one PPT task. The PPT is always active and remains active even if VTAM terminates. Termination of the PPT task abends NetView.

Because the PPT must always be present, it can be used for running certain commands that must always run. An example of such a command is one for controlling the scheduling of timers.)

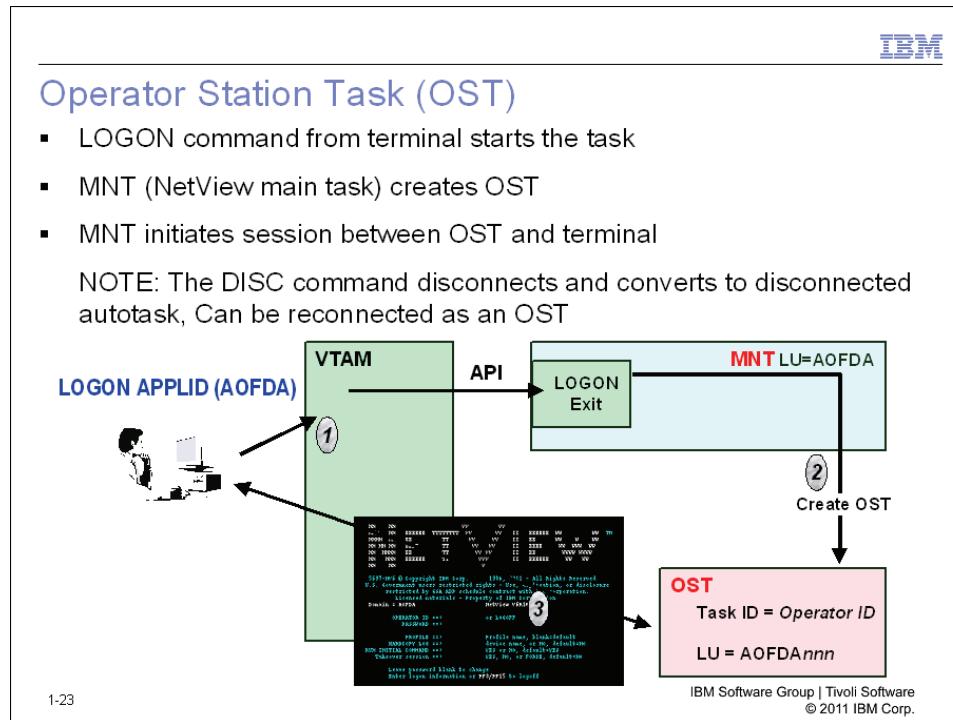
The PPT can receive and process messages, run commands and EXECs, and run timer-driven requests. The PPT plays a major role in the routing of VTAM messages. It cannot directly display messages, but can perform some automation functions, such as running EXECs



Important: Be careful of the work that you schedule under the PPT task:

- Not all functions are supported under the PPT. For example, the PPT does not support waiting or trapping messages.
- Schedule automation to run under other tasks, such as automation tasks to keep the PPT task free to process messages and timers. Such strategy can help you avoid possible performance problems.

Operator Station Task (OST)



The NetView main task, DSIMNT, creates an Operator Station Task (OST) for each operator who logs on to NetView. An OST is an *attended* task. In VTAM terms, the OST pertains to the application (PLU) being in session with the terminal (SLU). If several operators are logged on, each operator can issue commands that run independently of each other. The execution is managed by the multitasking capabilities of z/OS.

- OST routines check operator authorization at logon and analyze commands that invoke appropriate processors. OSTs end at operator logoff or when NetView or VTAM terminates.
- The DISC command converts an OST into a *disconnected* autotask, which can be later reconnected or taken over as an OST again.
- OSTs can receive and display messages, run commands, run timer-driven requests; and run full-screen commands. All NetView commands must run within an OST.

Automated Operator Task (AOST)



Automated Operator Task (AOST)

- Automated operator task (AOST) flow similar to OST
 - Does not support panels
 - Does not require VTAM
- Started with AUTOTASK command
AUTOTASK OPID=AUTO1, CONSOLE=*ANY*
- Most common task used in NetView automation
Target for many automation commands
- Also known as the following terms:
 - Automated operator
 - Automation task
 - Automated operator task
 - Autotask

1-24

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Automation tasks (Autotasks) are OSTs that are used for *unattended* operations. An autotask is an OST that does not have a logged-on operator directly associated with it. An autotask has similar characteristics to an OST. It can receive (but not display) messages. It can run commands. It can run timer-driven commands. It cannot execute full-screen commands such as HELP, NLDM, or NPDA.

Autotasks can start during NetView initialization (using CNMSTYLE) or with the AUTOTASK command. An example is as follows:

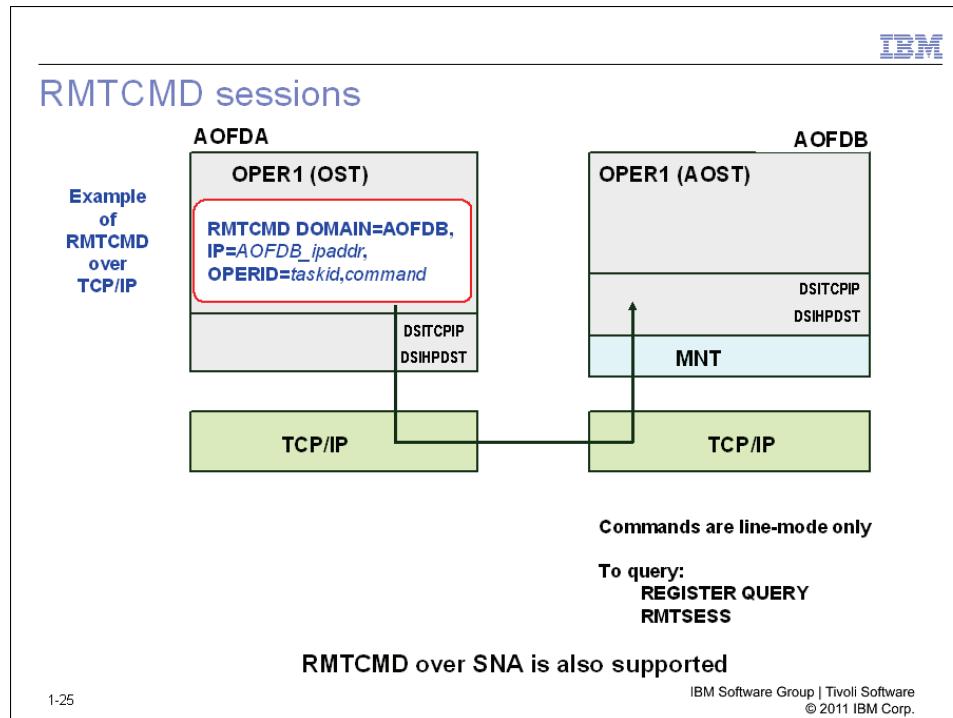
```
AUTOTASK OPID=AUTO1 CONSOLE=*ANY*
```

An autotask can start without requiring VTAM to be active. It remains active even after VTAM termination. Autotasks can start after a NetView CLOSE command has been issued, usable for automating shutdown. After VTAM is active, the autotask uses VTAM access control block (ACB). You define your VTAM APPL definitions to accommodate operators and autotasks.

If NetView commands are entered from an EMCS console, use the optional CONSOLE operand to indicate that those commands are to run in this autotask. (All commands or EXECs must execute in an OST task.) Specify the console name. Use console names *MASTER* and *ANY*. If a console does not have an associated autotask, NetView commands cannot be entered from it.

For more details on console names, see the online help for the AUTOTASK command or the *IBM Tivoli NetView for z/OS Administration Reference* manual.

RMTCMD sessions



Use the RMTCMD command to send a command that is to run in another NetView domain. The originating domain receives responses to the command.

The remote domain might be connected by SNA LU 6.2 or TCP/IP (IPv4 or IPv6). With RMTCMD, you can send more data than by using the older NNT support.

If no remote task is in the RMTCMD command, an existing session to that remote domain is used. If none exists, a *distributed autotask* starts in the remote domain. The remote domain uses the same name as the originating operator, and this session becomes an owned session.

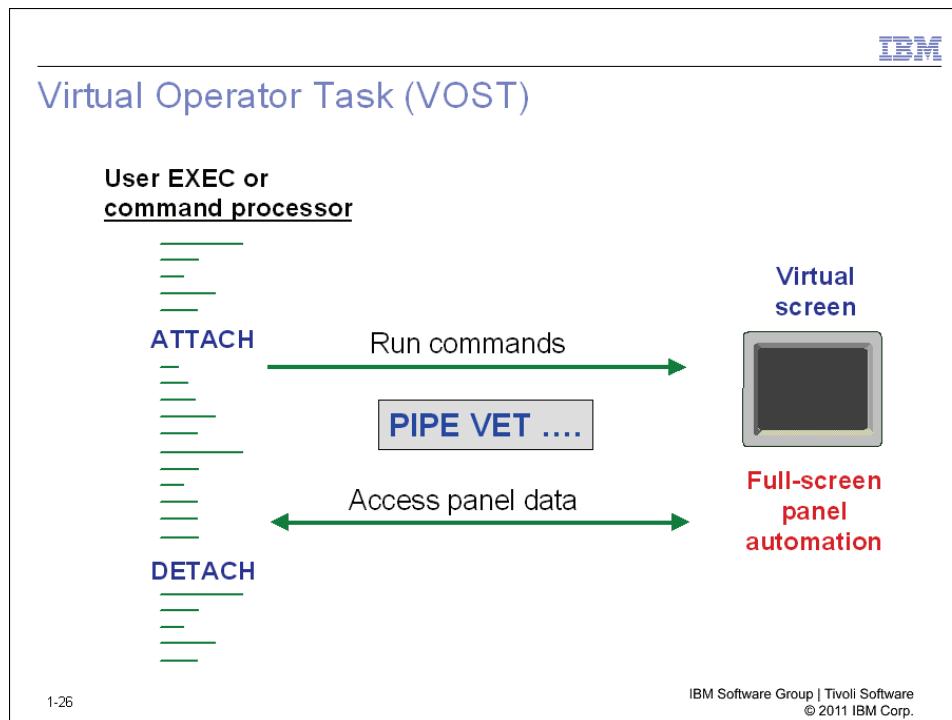
If a remote task is in the RMTCMD command, and the task already exists on the remote domain, the command routes to the remote autotask. The session does not become an owned session. If the desired task does not exist, a *distributed autotask* with the desired name starts in the remote domain, and this session becomes an owned session.

Security can be enhanced when using the RMTCMD command. (See LIST DSUDST command and the *NetView Security Reference* manual).

The RMTSESS command reports existing RMTCMD owned sessions. The REGISTER QUERY command reports the state of owned RMTCMD sessions. You can end owned sessions by issuing the ENDTASK command.

Always use the RMTCMD command if using EMCS consoles, not NNT connections. The full set of message characteristics return with each message. Commands that display a panel are not supported.

Virtual Operator Task (VOST)



Use a Virtual Operator Station Task (VOST) to automate full-screen 3270 sessions, sometimes called *screen scraping* automation. A VOST is an *unattended* task.

Create a VOST by issuing an ATTACH command (usually from an EXEC). The VOST associates with the task (OST or AOST) that creates it.

The VOST logs on to 3270 applications, usually through a TAF session. The VOST maintains a virtual screen that reflects the current state of the session. This virtual screen can be made available to the creating task. The creating task can enter data on the virtual screen and request that the ENTER and PF keys be pressed. The creating task can also request the current contents of the screen to analyze the results of the commands entered.

The DETACH command releases the VOST. Whether the VOST ends at OST logoff or not depends on how the VOST is invoked.

VOST tasks can be displayed with the following command:

```
LIST STATUS=VOST
```

See the *Programming: PIPES* manual for further details.

LIST STATUS=TASKS

IBM

LIST STATUS=TASKS

TYPE	TASKID	RESOURCE	STATUS
MNT	MNT	AOFDA	ACTIVE
PPT	AOFDAPPT	AOFDAPPT	ACTIVE
OPT	DSILOGMT	DSILOGMT	ACTIVE
OPT	DSILOG	DSILOG	ACTIVE
OST	NETOP1	A01A701	ACTIVE
OST	AUTDVIPA	AUTDVIPA	ACTIVE
OST	AUTOAON	AUTOAON	ACTIVE
OST	AUTO1	AUTO1	ACTIVE
OST	AUTO2	AUTO2	ACTIVE
OPT	AOFDABRW	AOFDABRW	ACTIVE
OPT	AOFDALUC	AOFDALUC	ACTIVE
OPT	AOFDASIR	AOFDASIR	ACTIVE
OPT	BNJDSERV	BNJDSERV	ACTIVE
OPT	DSICRT	DSICRT	ACTIVE
OPT	DSISVRT	DSISVRT	ACTIVE
OPT	DSITCP	DSITCP	ACTIVE
OPT	DSIUDST	DSIUDST	ACTIVE
OPT	DSIHPDST	DSIHPDST	ACTIVE
	DSIWBTSK	DSIWBTSK	ACTIVE
	DSI6DST	DSI6DST	ACTIVE

1-27

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide shows an example of a LIST STATUS=TASKS command to display all tasks that are currently defined to NetView. The resource name for the main task indicates that the command was entered from NetView AOFDA domain. The PPT task name is the NetView domain ID concatenated with the PPT character string. In this case, it is AOFDAPPT.

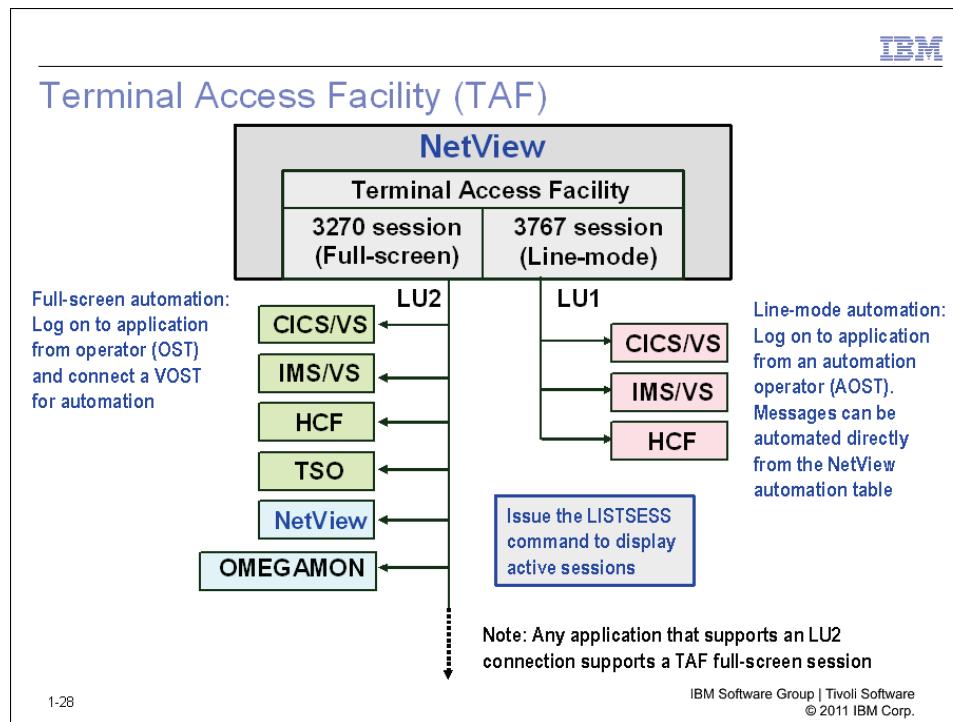
Several tasks as follows are active on AOFDA domain:

- NETOP1 is an OST logged on from the A01A701 terminal.
- AUTO1 is an automation task (AOST). You can determine that because the resource (terminal) name is the same as the task name.
- AUTO2 is also an automation task.
- NPDA is active. BNJDSERV is listed as one of the optional tasks (OPT).
- DSITCP, DSIUDST, and DSIHPDST tasks are active. This means that the AOFDA domain is enabled for RMTCMD sessions over TCP/IP and SNA.



Note: LIST STATUS=TASKS might also display tasks that are defined to NetView but not currently active. The example shows only active tasks.

Terminal Access Facility



The Terminal Access Facility (TAF) of NetView simulates a terminal logon to a chosen VTAM application. The NetView operator is then logged on to both NetView and target application simultaneously. A target session might run in line-by-line mode (LU1 session type, simulating a 3767 terminal), or in full screen mode (LU2 session type, simulating a 3270 terminal). You can use a TAF to log on to any VTAM application that supports LU1 or LU2 sessions, including a remote NetView domain.

A TAF full-screen (FLSCN, 3270 LU2) session cannot be used for message automation. A TAF operator-control mode (OPCTL, 3767 LU1) session uses line-by-line mode. This type of session supports message exchange between NetView and the target application providing a mechanism for automation of messages from the application.

An autotask can use the TAF operator-control mechanism to manage CICS/ VS and IMS/VS sessions. You can also use the NetView PPI to manage CICS and IMS.



Tip: Use a VOST for full-screen automation of TAF LU2 sessions.

Other major tasks

IBM

Other major tasks

- DSICRTR: For forwarding CNMI data and alerts

```
graph LR; VTAM[VTAM  
CNMI] --> CRTR[CRTR]; CRTR --> NPDA[NPDA]; CRTR --> NLDM[NLDM]
```

- CNMCSSIR: For z/OS SSI communication

```
graph LR; zOS[z/OS  
SSI] --> NetViewSSI[NetView  
SSI]; NetViewSSI --> NetViewSSIR[NetView  
SSIR]
```

1-29

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide shows the Communication Network Management Router Task (DSICRTR) and Subsystem Interface Router Task (CNMCSSIR).

The CRTR handles data and alerts that arrive from the VTAM CNMI. The CRTR forwards CMN data from VTAM to tasks, such as NPDA and NLDM, as necessary. Alerts are sent to the Hardware Monitor database.

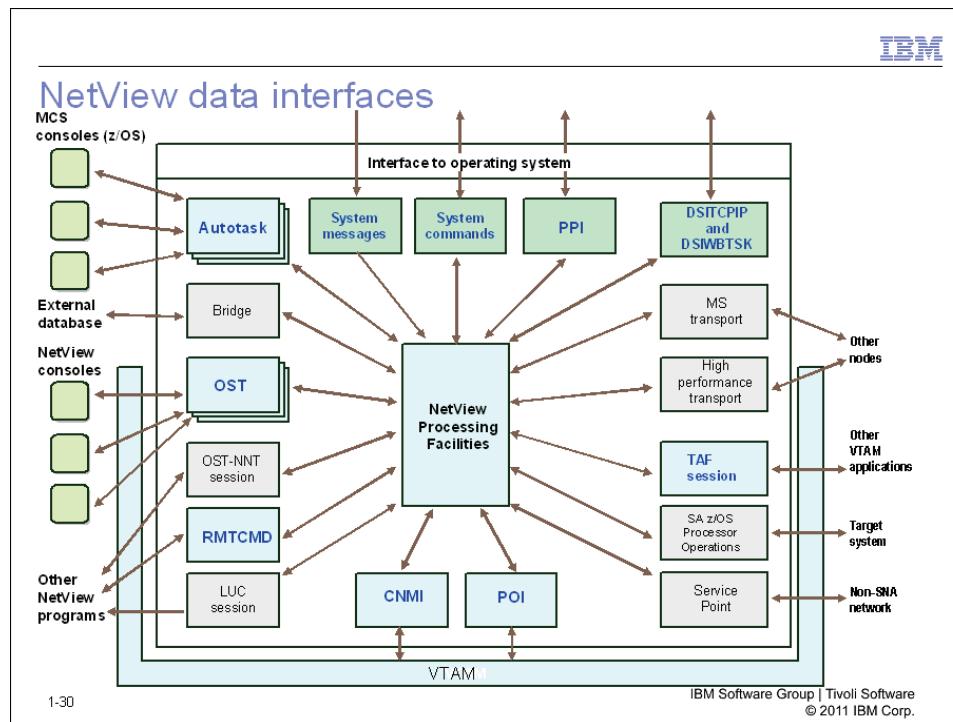
The CRTR task is optional. A NetView application can have only one CRTR task. If several NetViews are active within a system, each can have an active CRTR task. Only one can receive CNM data from VTAM, but each can be used for processing alerts.

The CRTR task is defined in CNMSTYLE and requires a corresponding VTAM APPL definition in order to forward CNMI data.

The SSIR task receives the flow of unsolicited messages from z/OS. NetView can be set up so that this flow comes through the z/OS SSI using the NetView subsystem address space. The flow comes directly from z/OS that uses the EMCS feature of z/OS. In effect, the SSIR task is defined to z/OS MCS as a console that receives messages with all routing codes.

Only one SSIR task is in each NetView application address space. The SSIR also handles commands that are entered at an EMCS console by using a command prefix. The SSIR then routes these commands to the autotask that is associated with that console (if any) for actual execution. Any output from such commands routes back to the originating EMCS console.

NetView data interfaces



This slide summarizes several interfaces that can present data to NetView. See the *Automation Guide* manual for full descriptions of these interfaces.

IBM

Student exercise



1-31

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your Student Exercises book and perform the exercises for this unit.

Summary



Summary

Now that you have completed this unit, you can perform the following tasks:

- Define passive versus active monitoring
- Define system versus network automation
- Describe single-system and multisystem automation
- Identify event types that can be processed and how they arrive in NetView

1-32

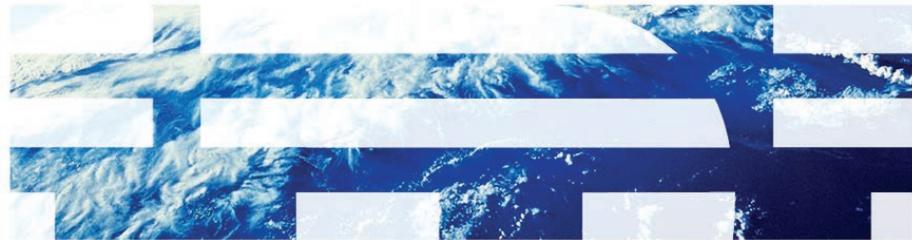
IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Unit 2: Message automation topics

IBM

Unit 2:

Message automation topics



© 2011 IBM Corp.

Introduction

This unit discusses the automation of messages with NetView. For example, using the message processing facility (MPF) of z/OS, you can suppress messages. The message revision table (MRT) of NetView provides functions similar to the MPF. This unit also provides information about the *primary receiver* for messages and how it pertains to message automation.

Objectives

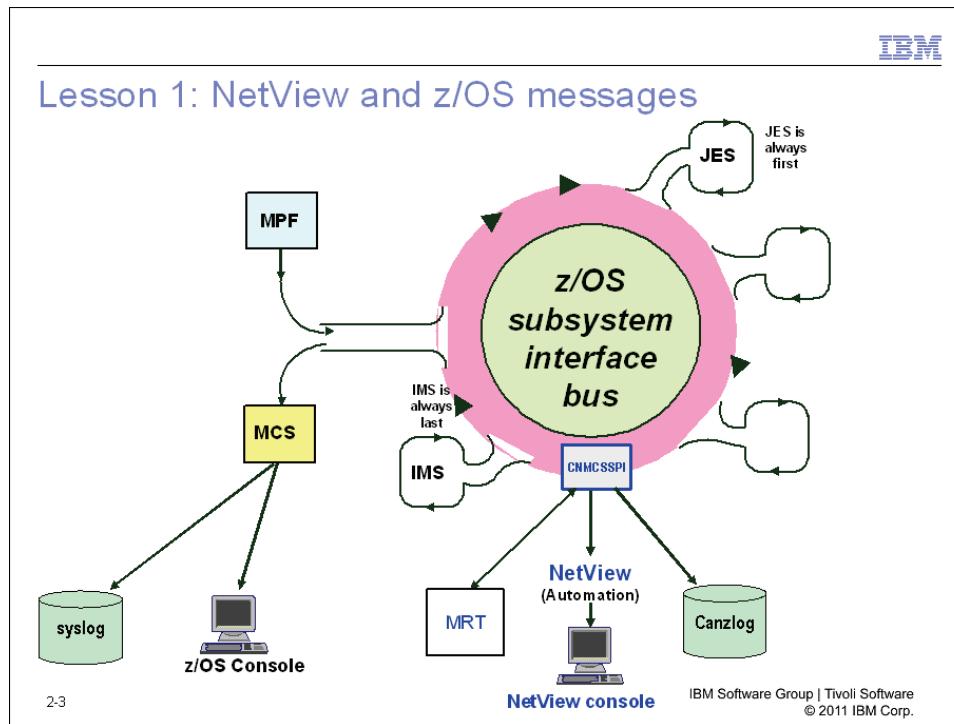


Objectives

When you complete this unit, you can perform the following tasks:

- Describe the functions provided by the message processing facility (MPF)
- Describe the functions provided by the message revision table (MRT)
- Assign a primary receiver for unsolicited messages

Lesson 1: NetView and z/OS messages



NetView is the base component for mainframe-based management and automation of system and network resources in your enterprise.

Messages can arrive from operating systems, their applications and subsystems, and also from the network. NetView analyzes messages, which pertain to the status of tasks and resources, then it takes actions as assigned by the installation.

When implementing an automation project, it is necessary to understand how NetView receives the relevant messages. You must also know how to work with the appropriate NetView facility to achieve the desired automation process. The networking connections where messages flow vary, requiring careful implementation and operation.

The slide shows one possible flow for receiving z/OS messages, the *z/OS subsystem interface bus*. Messages flow through MPF, then to the SSI, where alterations can be made that affect their routing and presentation. As part of SSI processing, messages are written to the NetView Canzlog data space. After the SSI, messages are normally routed through MCS to one or more consoles. NetView is one of many application on the SSI bus. JES is usually the first application on the SSI bus, and IMS is usually the last. The order of the remaining applications is based on the order of their subsystem names that are defined in PARMLIB member, IEFSSNxx.

The NetView listener is named CNMCSSPI, actually in the LPA, but logically, in the SSI bus.

If your installation produces large volumes of messages for operators to monitor, use the action message retention facility (AMRF). If you want operators to be able to retrieve action messages and WTOR messages that are no longer on the console, use AMRF. AMRF keeps action messages so that the operator can view them later. WTOR messages are always available for operator retrieval regardless of the state of AMRF.

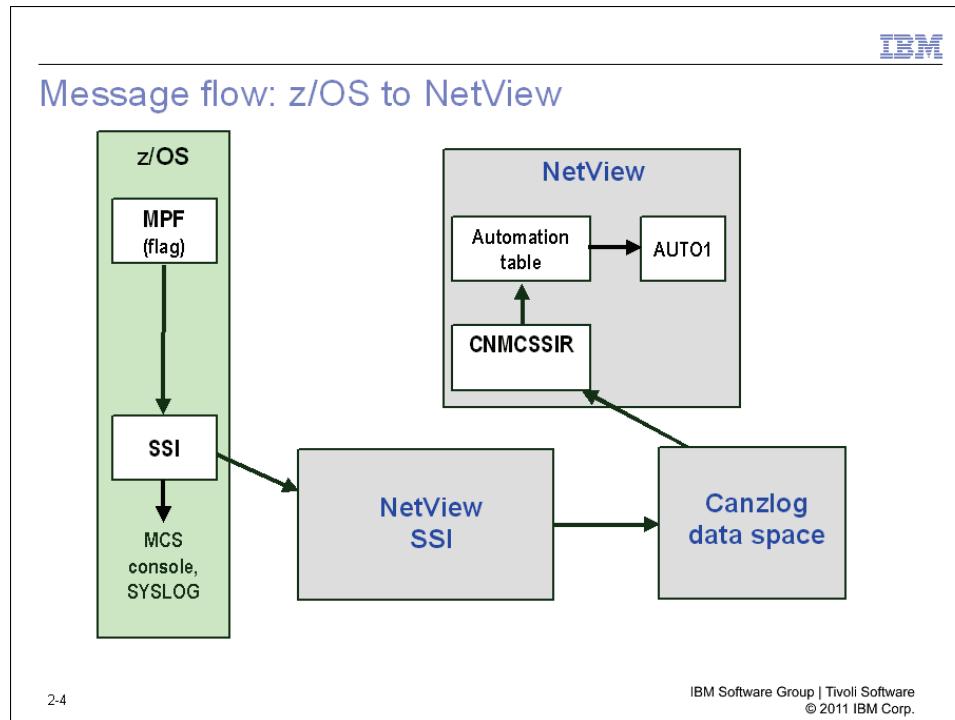
When the operator has performed the action required that a screen message displays, the system deletes the message, or the operator can use the CONTROL C command to delete the message. If AMRF is active, operators can remove action messages from the screen, then retrieve them in their entirety later by using the DISPLAY R command.

Use the MVS DISPLAY SSI command to display all of the SSI applications and their attributes, for example:

```
IEFJ100I 04.20.46 SSI DISPLAY 537
SUBSYS=JES2 (PRIMARY)
    DYNAMIC=YES      STATUS=ACTIVE      COMMANDS=REJECT
SUBSYS=MSTR
    DYNAMIC=NO       STATUS=ACTIVE      COMMANDS=N/A
SUBSYS=SMS
    DYNAMIC=YES      STATUS=ACTIVE      COMMANDS=REJECT
SUBSYS=AUTO
    DYNAMIC=YES      STATUS=ACTIVE      COMMANDS=ACCEPT
SUBSYS=EKGX
    DYNAMIC=YES      STATUS=INACTIVE    COMMANDS=REJECT
SUBSYS=RACF
    DYNAMIC=YES      STATUS=ACTIVE      COMMANDS=REJECT
SUBSYS=DFRM
    DYNAMIC=YES      STATUS=INACTIVE    COMMANDS=REJECT
SUBSYS=TNF
    DYNAMIC=YES      STATUS=INACTIVE    COMMANDS=REJECT
SUBSYS=VMCF
    DYNAMIC=YES      STATUS=INACTIVE    COMMANDS=REJECT
SUBSYS=BLX1
    DYNAMIC=YES      STATUS=INACTIVE    COMMANDS=REJECT
SUBSYS=CNDL
    DYNAMIC=YES      STATUS=INACTIVE    COMMANDS=REJECT
SUBSYS=AXR
    DYNAMIC=YES      STATUS=ACTIVE      COMMANDS=REJECT
```

In this example, AUTO is the subsystem name for the NetView application (AUTONETV) and NetView SSI Assist Procedure (AUTOSSI). If you need the NetView SSI to be inserted into the z/OS SSI bus earlier or later, you move the subsystem name definition. Issue the MVS SETSSI command to dynamically update the subsystem name table.

Message flow: z/OS to NetView



NetView can receive messages from the z/OS operating system. The messages pass from z/OS to NetView as follows:

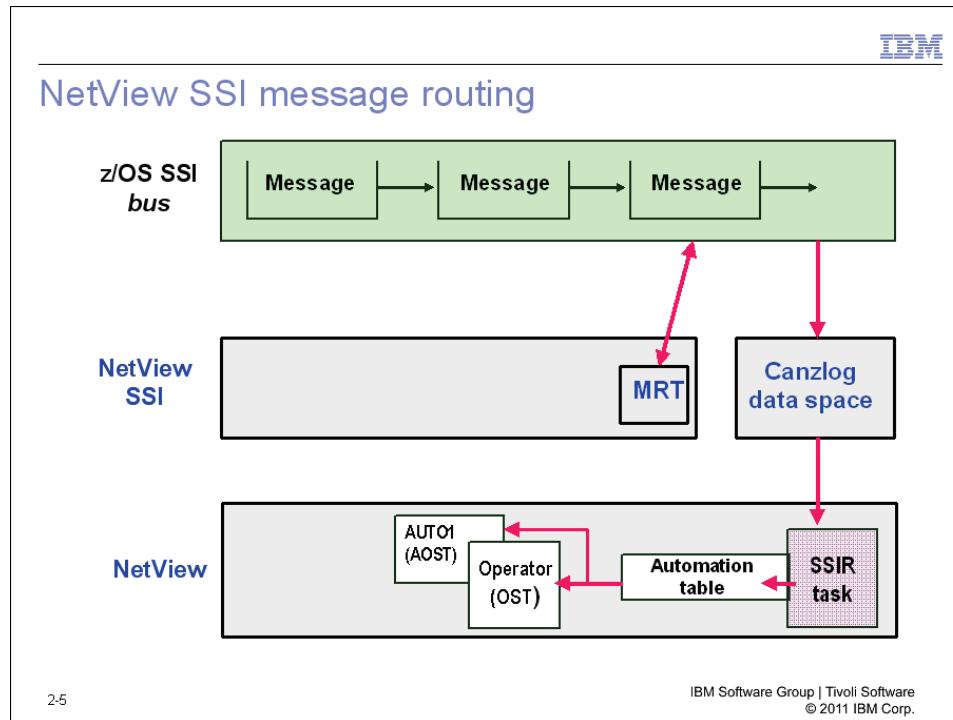
1. The z/OS application issues a WTO macro, creating the z/OS message.
2. The z/OS message processing facility (MPF) examines the message, and sets the automation and suppress flags according to the MPF list specification.
3. The Multiple Console Support facility passes the message through the active SSI exits. The NetView exit (or exits) pass the message through MRT processing (if active), and copies the message with attributes into the Canzlog data space.
4. After MPF and MRT processing, the message might or might not be wanted for automation. If it is, the NetView SSI exit posts the NetView task CNMCSSIR to find new messages. The original message, with any changes made by the MRT, continues in parallel with z/OS consoles and syslog, as appropriate.
5. The CNMCSSIR task extracts the next automatable message from the Canzlog data space, and routes it through the automation table.



Note: Messages that are directed to an EMCS console that a NetView task owns are automated by that task. Other messages are automated at the CNMCSSIR task.

When the CNMCSSIR task is recycled by the RECYCLET command or is a result of RESTYLE MVSPARM command, no messages are bypassed by automation. However, when NetView itself is recycled or if the CNMCSSIR task is down, the new style statement, MVSPARM.Msg.Automation.MaxAge, controls how old messages can be and still be automated. The MaxAge value can be set as high as 86400, one full day, or as low as zero.

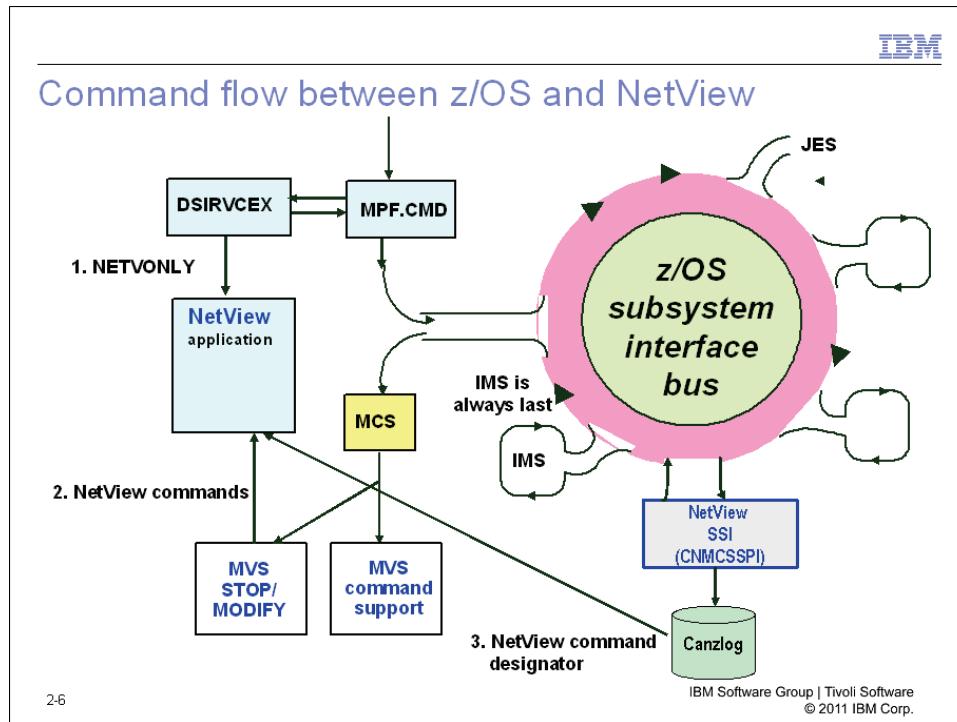
NetView SSI message routing



NetView SSI message routing occurs as follows:

1. The CNMCSSIR SSI router task, which runs in the NetView application address space, routes messages and commands between the NetView SSI and the NetView application.
2. The SSIR task inspects the buffers in the Canzlog data space, and copies (and queues) selected messages to process in the NetView application address space.
3. When the messages arrive, they route to the various NetView automation facilities, including the automation table. These, in turn, might cause the messages to route to one, some, or none of the different OSTs in NetView.

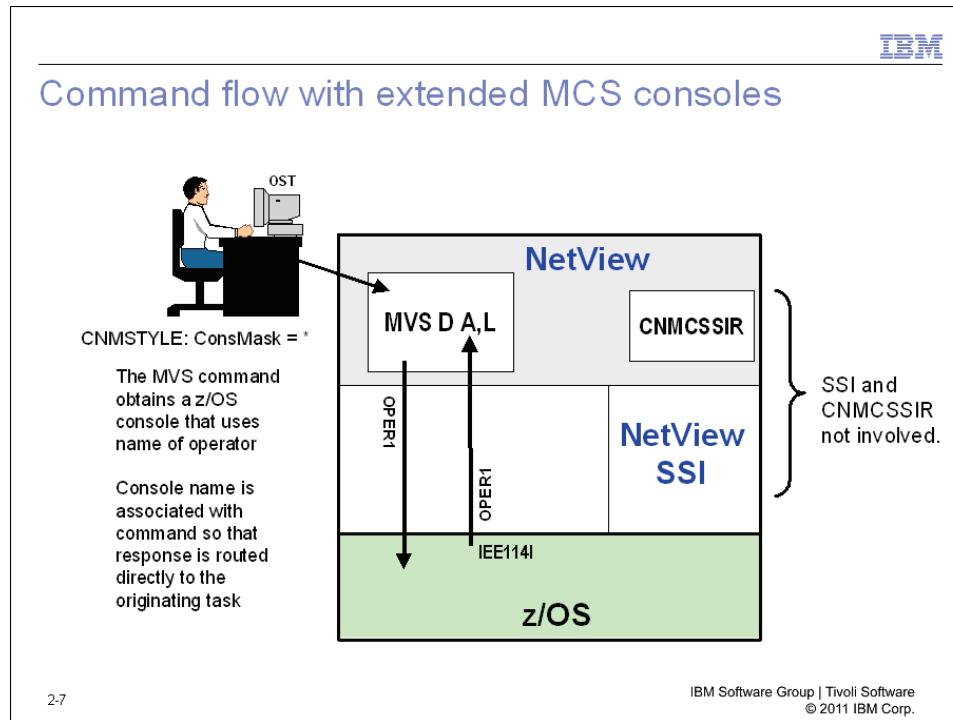
Command flow between z/OS and NetView



This slide shows the flow of commands between z/OS and NetView, as summarized as follows:

1. Command Revision Table might revise commands. Commands that are flagged with NETONLY are sent to the NetView application.
2. Commands that are sent to NetView through a MODIFY command are retrieved directly from z/OS.
3. Commands with the NetView command designator are routed through the Canzlog data space.

Command flow with extended MCS consoles



In this slide example, the MVS command obtains an Extended MCS console if the name matches the task name. This console name is sent with the command to z/OS. The command response is created using the WTO macro, but in this case, the MCS directly passes the response back to the calling task. The NetView subsystem and the SSIR task are not involved.

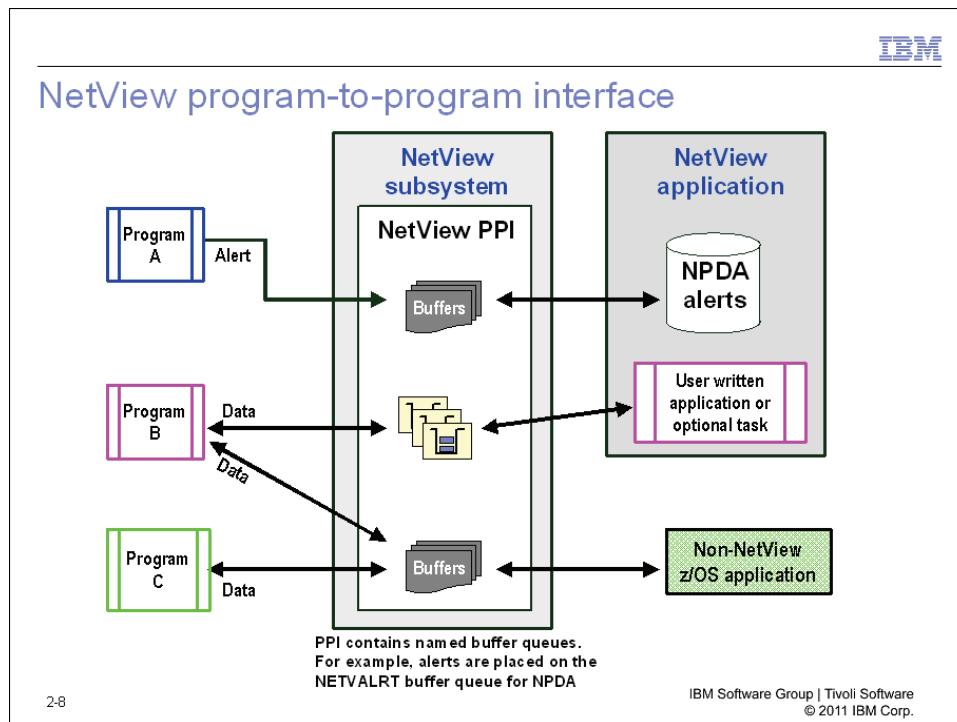


Note: Console names within a sysplex must be unique. You can use the *ConsMask* statement in CNMSTYLE to create unique console names.

In this example, **ConsMask = *** was defined to use the operator ID as the console name.

You can issue the GETCONTD command to obtain the console before any MVS command runs. This action helps because a sysplex might have several NetView applications, each with similar task names. Use GETCOND to create unique task names for each application. If you use a SETCONID command, you can set a name without actually creating an EMCS console. This is useful when some NetView tasks do not issue MVS commands.

NetView program-to-program interface



The NetView SSI address space supports the NetView program-to-program interface (PPI) for transfer of data between applications. NetView can be one of the applications.

Any application program that runs under z/OS can send data to the NetView PPI by using z/OS *cross memory services*. The data goes into named buffer queues. With PPI, programs can pass data directly to any other program in the same system. NetView applications can be one, both, or none of the programs involved in such transmissions. Note examples as follows:

- Use of the PPI to send messages from one batch job to another
- Use of the NetView PPI buffer called NETVALRT for sending alerts to NetView
- Applications that use the NetView PPI to share data. The applications can be in the NetView address space or in separate z/OS address spaces. NetView macros are used for sending and receiving data. For more information, see the *NetView Application Programming Guide*.

Automation in a sysplex environment



Automation in a sysplex environment

- Sysplex containing multiple z/OS systems
- Managed as a single system
 - Applications can be active on any system within the sysplex.
 - Allows sharing of data, for example.
- Cross-system coupling facility (XCF) required
- Automation by NetView in each system
 - Multisystem automation
 - One NetView: Focal Point or Single Point of Control (SPOC)

2.9

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

A z/OS sysplex is a configuration of multiple z/OS operating systems that work as a single system by sharing functions and programs. A key component is the cross-system coupling facility (XCF). When using NetView automation in a sysplex environment, NetView can use the standard z/OS commands. These commands include routing commands to other systems in the sysplex and receiving their responses. The following considerations are significant:

- Using Extended MCS consoles
- Coordinating z/OS message processing options with Extended MCS
- Deciding whether to centralize system automation on one system or not

For more information pertaining to sysplex automation, see the *NetView Automation Guide*.



Tip: SA z/OS provides you with extensive automation and management capabilities for your sysplex.

Extended MCS consoles in a sysplex



Extended MCS consoles in a sysplex

- You use extended MCS consoles (not SSI)
- You need a naming convention for system consoles
Unique console names across sysplex
- CNMCSSIR acquires extended MCS console
 - SSIR task needs a unique console name
 - For example, AOFDASIR, where AOFDA is the NetView domain
- Messages process on one system
Can be passed to NetView on another system
- Only R8 and later releases of z/OS support console names

2-10

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Use Extended MCS consoles for message delivery instead of using the SSI when using Netview for automation in a sysplex. A strategy is necessary for naming consoles.

EMCS console names must be unique across the sysplex. Use the *ConsMask* statement in CNMSTYLE to create unique console names. If you assign console names to match the task names, and the CNMCSSIR task acquires an EMCS console, ensure a unique name for each CNMCSSIR task. The convention is to create a name of the form *xxxxxSIR*, where *xxxxx* is the NetView domain name. An example is AOFDASIR.

Operators might be logged on to several NetView applications in the same sysplex while also using TSO SDSF to enter z/OS commands. If so, they need to obtain consoles with unique names across the sysplex.

The MPF table in the system that originated a message processes the message. However, the message might pass to other systems where a NetView facility might perform further processing.

Console management commands



Console management commands

Four commands to manage MVS consoles:

- GETCONID: Acquires extended MCS console
Console is also acquired with first MVS command
- SETCONID: Associates a task with a console
 - Does not allocate the console to the task
 - You use SETCONID during initialization for defining unique console names and associating with each task
- RELCONID: Releases any acquired MVS consoles
Logoff also releases console
- DISCONID: Displays all MVS consoles that are acquired

2-11

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

By default, when you issue an MVS command (for example, MVS D A,L), you acquire an EMCS console name as the task name that issues the command. This situation can cause problems in a sysplex environment because the console names must be unique across a sysplex.

Using a naming strategy, you can achieve unique console names with GETCONID. Acquire a unique EMCS console (or issue a SETCONID command to associate a unique EMCS console) in an operator initial command list (IC). The initial command list is run when the operator logs on to NetView.

The operator initial command list is defined in the operator profile. For example, suppose you use profile DSIPROFB and it looks like the following text:

```
DSIPROFB      PROFILE  IC=LOGPROF1
                AUTH     MSGRECVR=YES , CTL=GLOBAL , NGMFADMN=YES
                END
```

In this case, the operator initial command list is LOGPROF1. Each NetView task can have only one EMCS console associated with it.

EMCS example

The screenshot shows a NetView log window titled "EMCS example". The log output is as follows:

```
NetView V6R1MO      Tivoli NetView    AOFDA TSCCW01 07/08/11 05:36:15
  * AOFDA   SETCONID CONSOLE=MYCONS
  - AOFDA   DS1633I SETCONID COMMAND SUCCESSFULLY COMPLETED
  ' AOFDA   DISCONID

  1 CNM492I OPERATOR ID    CONSOLE ID    CONSOLE NAME
  CNM492I -----
  CNM492I AUTOAON        EXTENDED       AUTNOAAD
  CNM492I AUTO1          0 * *MASTER*
  CNM492I END DISPLAY

  2 CNM492I MVS D T
  E AOFDA   IEEE1361 LOCAL: TIME=05.36.09 DATE=2011.189 UTC: TIME=10.36.09
  DATE=2011.189
  * AOFDA   DISCONID

  3 CNM492I OPERATOR ID    CONSOLE ID    CONSOLE NAME
  CNM492I -----
  CNM492I TSCCW01        EXTENDED       MYCONS
  CNM492I AUTOAON        EXTENDED       AUTNOAAD
  CNM492I AUTO1          0 * *MASTER*
  CNM492I END DISPLAY
```

Annotations are numbered 1 through 4:

- Annotation 1: Points to the first line of the log output.
- Annotation 2: Points to the second line of the log output.
- Annotation 3: Points to the third line of the log output.
- Annotation 4: Points to the fourth line of the log output.

A callout box points from annotation 4 to a note: "After issuing the MVS D T command, TSCCW01 has an active console named MYCONS".

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

In this slide example, the console named MYCONS associates with operator (OST) TCCW01 from the use of the SETCONID command. The DISCONID command shows three consoles have been allocated. MYCONS is not allocated because no commands have been issued yet. The MVS D T command is issued. This causes the MYCONS console to be acquired by TSCCW01, as shown in the second DISCONID command.

MCS and message dispersal

IBM

MCS and message dispersal

```
graph LR; A[Messages] --> B[MCS]; B --> C[Computer Monitor]; B --> D[Computer Monitor]
```

- Reduces traffic down to a particular console
- Does not reduce overall traffic
- Can classify messages
Route to consoles using code type as follows:
 - Descriptor
 - Route

Route codes and descriptor codes are discussed in the *MVS Programming: Assembler Services Reference, Volume 2* manual, under the WTO macro description.

2-13

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can reduce message traffic to a particular console. MCS enables specific consoles to receive particular *classes of messages*. Total traffic does not reduce, but message dispersion improves.



Note: Within NetView, the ASSIGN command and automation table also perform a similar routing function, to be discussed in a later unit.

Each z/OS system message has two codes so that you can define the message flow, enabling message processing by the operators as follows:

- Descriptor code: Provides a description of the significance of a message, used by MCS for determining how the message is to be displayed.
- Route code: Provides the ability to group messages by function, used for routing the message to the appropriate MCS consoles.

Descriptor codes for z/OS messages



Descriptor codes for z/OS messages

- Descriptor codes usable for processing messages
 - Identifies message significance
For example, wait for operator action
 - Holds the message on the console
 - Identifies the type of message
For example, critical eventual action required
- Examples
 - 1: System failure: Operator must IPL system or restart a major subsystem
 - 2: Immediate action necessary: Program waits
 - And so on
- Automation table and REXX able to investigate descriptor code DESC() function

2-14

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

A z/OS message might have a descriptor code (numbered from one to 16) associated with it. The descriptor code describes the significance of the message (for example, immediate action required). The code designates how the system displays the message, or if it deletes the message.

The last character of a message ID, by convention, matches the descriptor code. For example, a last character of A (as in DSI802A) indicates an immediate-action message.

Examples of descriptor code values are as follows:

- 1 = System failure
- 2 = Immediate action required

Descriptor codes of 1, 2, 3, and 11 denote *action* messages. Action messages stay on the MCS console display until removal, either by an operator or because of programming. Action messages can optionally be retained by the Action Message Retention Function (AMRF) for later reviewing. The exception is the active Message Processing Facility specification that indicates otherwise. Action messages also stay on NetView consoles.

Route codes for z/OS messages



Route codes for z/OS messages

- Route codes usable for processing messages
 - Group messages by function
 - Route message to specific consoles
- Examples
 - 1: Message to master console for action
 - 2: Message to master console for information
 - 3: Tape pool message
 - 4: Security system message
 - And so on
- Automation table and REXX able to investigate route code
ROUTCDE() function

2-15

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Route codes (numbered from one to 128) are the key to using MCS for a z/OS environment. With route codes, you can group messages by function, and send messages to specific consoles. Examples of route code values are as follows:

- 1 = Message to master console for action
- 2 = Message to master console for information
- 3 = Tape pool message
- 9 = System security message

For example, an MCS Console could be defined to receive only messages of route codes one, two, and nine, but not messages with other routing codes. A message might have more than one route code.

NetView automation table processing for messages depend on the descriptor codes, routing codes of messages, or both.

Message processing facility (MPF) for z/OS

IBM

Message processing facility (MPF) for z/OS

The MPF provides control of z/OS messages and commands:

- Message presentation
 - For example, color, highlight, and intensity
- Message management
 - Suppression: Display or suppress
 - Retention: Retain in AMRF or not
 - Processing: Automate or not, or pass to user exit
- Command processing
 - User exit gets control each time a command is issued

```
graph LR; A[Commands and messages from z/OS] --> B[MPF]; B -- Process --> C; B -- Present --> D; B -- Manage --> E;
```

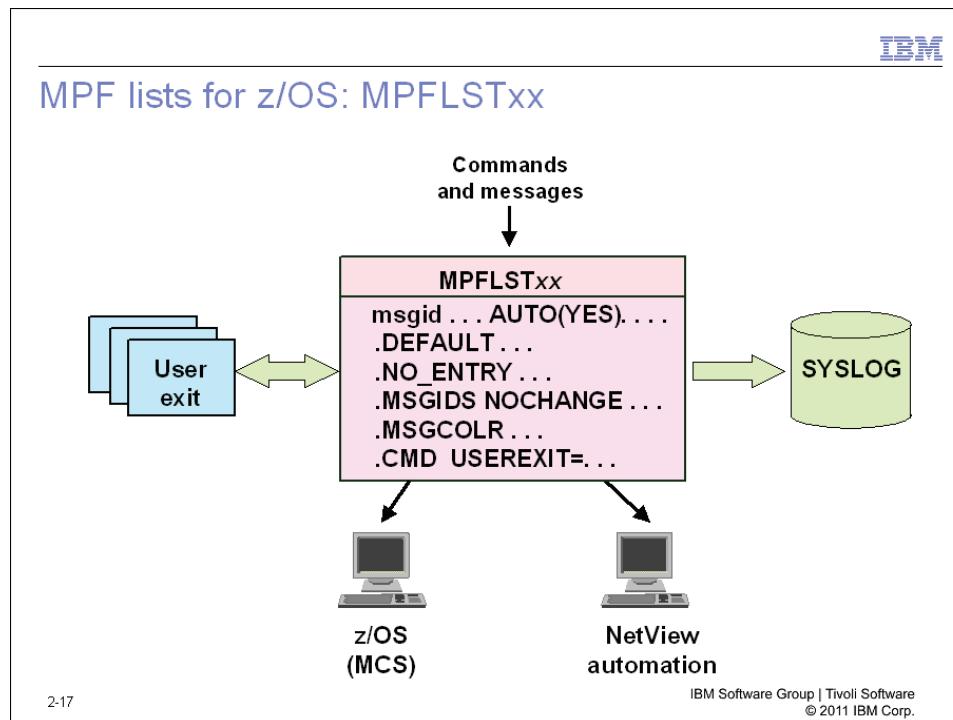
2-16

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

In a z/OS environment, the message processing facility (MPF) controls as follows:

- Command processing
 - Provides control to user each time a command is issued. This feature is available if the installation defines user exits that receive control when a command is entered. The NetView Command Revision Table (CRT) uses this z/OS feature to screen commands issued by all operators. NetView Command Revision Table discussion is in Unit 3.
- Message presentation
 - Determines color and intensity of messages on operator consoles. Highlighting indicates that the messages are in color.
- Message management
 - Message suppression: Indicates if messages are to be displayed on an MCS console where they are directed. WTO(R) messages are always written to the syslog.
 - Message retention: Indicates if messages are retained by automation message retention facility (AMRF).
 - Message processing: Indicates messages that are to be processed by a message automation subsystem (for example, NetView), or by an MPF user exit.

MPF lists for z/OS: MPFLSTxx



Message processing facility lists define how MPF processes commands and messages. The lists are MPFLSTxx members in your PARMLIB. You can create as many MPF lists as you want. The most common use of MPF lists is for *message management*, with some *message presentation*, and sometimes with *command processing*.



Note: IBM does not supply any MPFLSTxx members with the z/OS products. SA z/OS generates an MPF list (MPFLSTSA) when you build the automation policy.

See the *z/OS Installation And Tuning Reference* manual for more detailed information.

MPF implementation and use

MPF implementation and use

- MPFLSTxx loaded from PARMLIB
 - CONSOLyy
 - COMMNDzz
- Also loaded with SET command
 - SET MPF=xx
 - SET MPF=(xx,yy,...) for concatenation
- SET MPF=NO: Use of only IBM-supplied defaults
All messages automated
- D MPF: Display of current settings

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

2-18

Various MPF members might be in the z/OS PARMLIB. Each member has a name of MPFLSTxx and is activated by one of these following methods:

- The specification of an MPFLSTxx member. The member is in the INIT statement of the CONSOLyy member of PARMLIB that is used at system initialization.
- Issuing of the SET MPF=xx command. The command comes from the COMMNDzz member of PARMLIB that is used at system initialization.

An authorized user can issue the z/OS SET MPF=xx command. To properly concatenate several MPF lists, you must use the SET MPF=(xx,xx,..) command.

Using the SET MPF command to set a list overrides any current active list, unless the NOCHANGE operand is used within the MPF list definition. In such cases, some of the definitions are left unchanged.

The default is SET MPF=NO. If the operator issues the following command, default values are used for message presentation and message management of all messages:

Display all messages, retain all action messages, and send all messages to automation.

Display the active MPF lists by using the z/OS D MPF command.

Overview of MPFLST statements



Overview of MPFLST statements

- MPFLST statements are as follows:
 - DEFAULT: Define default settings for a section (group) of MPFLST
 - NO_ENTRY: Define settings for message not specified in MPFLST
 - MSGIDS NOCHANGE
 - Msgid: Define settings for a message or message string
- If you do not specify an MPFLSTxx, defaults are as follows:
 - Display all messages: SUP(NO)
 - Retain all messages: RETAIN(YES)
 - Automate all messages: AUTO(YES)
- Not to contain embedded blanks

2-19

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The most important of the MPF statements for message processing are as follows:

```
.DEFAULT
.NO_ENTRY
.MSGIDS NOCHANGE
msgid
```

If no message-processing statements or parameters occur in the active MPF list, default actions are as follows:

```
Display all messages
Retain all action messages
Automate all messages
```

These defaults are the same as when no MPF list is active.

MPF .DEFAULT example

IBM

MPF .DEFAULT example

.DEFAULT,AUTO(YES)

IEF402I,SUP(NO)

IEF403I

IEF404I,AUTO(NO)

.DEFAULT,AUTO(YES),RETAIN(NO)

IEF405I

.....

MPF defaults:
AUTO(NO) SUP(YES) RETAIN(YES)

- Define MPF default to pass all messages to automation
- If IEF402I, then display, automate, retain
- If IEF403I, then suppress, automate, retain
- If IEF404I, then suppress, retain, do not automate
- Define new default to pass all messages to automation and do not retain them
- If IEF405I, then suppress, automate, do not retain

2-20

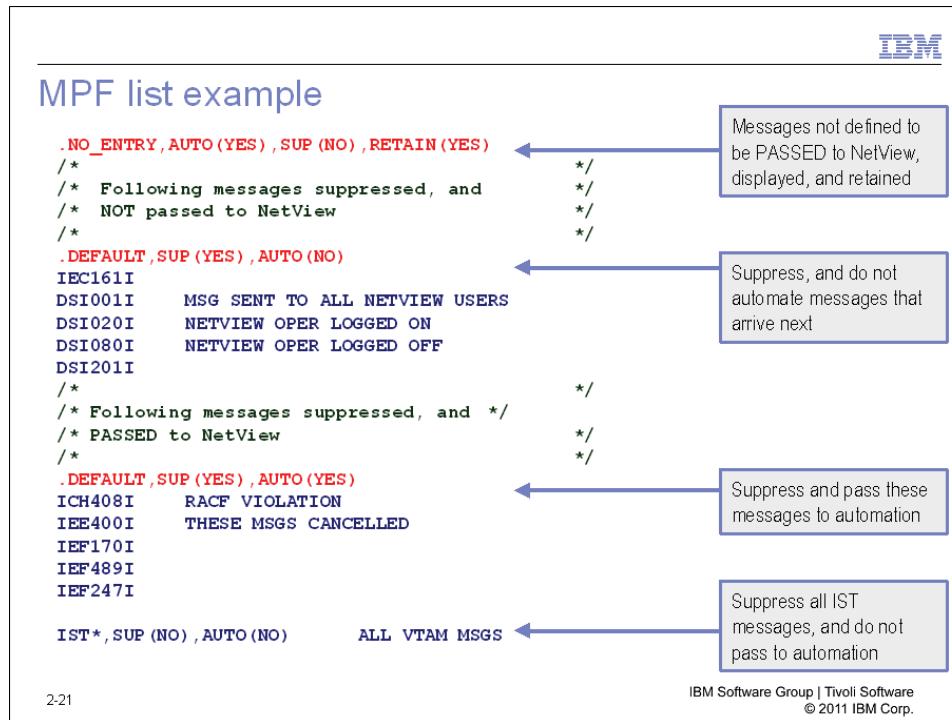
IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Any number of .DEFAULT statements can be coded. Any number of *msgid* statements that follow a .DEFAULT statement can override the values set on that .DEFAULT statement.

Four *msgid* statements are on the example slide: IEF402I, IEF403I, IEF404I, and IEF405I.

- The first .DEFAULT statement sets a default of AUTO(YES) for all messages that follow, and inherits the MPF default settings for SUP(YES) and RETAIN(YES). The example is IEF403I.
- Message IEF402I is displayed based on its SUP(NO), eligible for automation based on the .DEFAULT,AUTO(YES) statement, and retained based on the default MPF settings.
- Message IEF403I is eligible for automation based on the .DEFAULT,AUTO(YES) statement, and suppressed and retained based on the default MPF settings.
- Message IEF404I is not eligible for automation based on its AUTO(NO), and to be suppressed and retained based on the default MPF settings.
- The second .DEFAULT statement sets a default of AUTO(YES) and RETAIN(NO) for all messages that follow. The example is IEF405I.

MPF list example



This slide illustrates an example of an MPF list. The .NO_ENTRY statement specifies that messages that are not defined in the MPF list are to be sent to NetView for automation, shown, and retained. Messages that are not in the list are to be handled with the .NO_ENTRY statement.

The first .DEFAULT statement suppresses messages, and does not send them to NetView for automation. This applies to all of the messages that follow this .DEFAULT statement: IEC161I, DSI001I, DSI020I, DSI080I, and DSI201I.

The second .DEFAULT statement suppresses messages and sends them to NetView for automation. This action applies to all of the messages that follow this .DEFAULT statement: ICH408I, IEE400I, IEF170I, IEF489I, and IEF247I. The statement does not apply for all IST prefix (VTAM) messages. The IST* msgid statement overrides the second .DEFAULT statement by displaying the messages and not passing them to automation.

The .NO_ENTRY statement placed first. It is good practice to code one, even if it specifies the default values.

Lesson 2: Extended message management



Lesson 2: Extended message management

- Original z/OS-based messages route to SSI
Not copies
- Messages go through a message revision table (MRT):
 - Can revise as follows before presentation to the system log, console, or automation:
 - Color
 - Route code
 - Descriptor code
 - Message text
 - Display attributes
 - Syslog attributes
 - Automation (yes or no)
 - And so on
 - Possible actions:
 - Treating the same message differently, depending on its source
 - Suppressing message, deleting message, or sending it to automation only

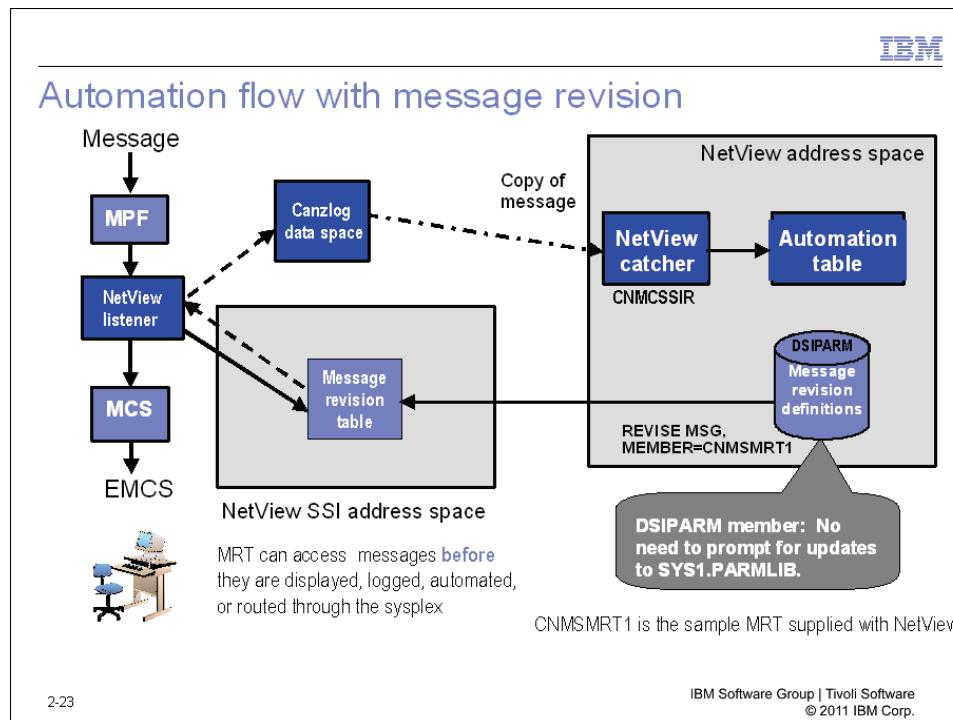
2-22

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This lesson discusses the functionality of the NetView message revision table (MRT). As messages come from the z/OS console into the NetView SSI, they can process through the NetView MRT. The MRT is in storage in the NetView SSI address space. You can use the MRT to suppress a message, change its attributes, and control whether it should be sent to NetView for automation or not.

The MRT can process any message from the z/OS console. They can be z/OS system messages, VTAM messages, application messages, and so on. Messages are received by the MRT after MPF processing but before being sent to a console. The MRT does not perform any automated functions. Automation of the message is still performed by the NetView automation table.

Automation flow with message revision



The MRT is a set of statements in DSIPARM. (The example member name is CNMSMRT1.) The NetView REVISE command loads the MRT in storage within the NetView SSI address space. Only one MRT for each NetView can be active. If you have three active NetView procedures (with different SSI names), each procedure can have its own MRT. Take care that the action of one MRT does not interfere with requirements of another. You can include members with %INCLUDE statements. Data REXX is also supported.

The *original z/OS messages* (z/OS system messages, VTAM messages, application messages, and so on) go to the NetView SSI, after processing by the message processing facility (MPF). If the message is suppressed with MPF, it goes to the SSI bus. Each application on the SSI bus can change the message attributes. For example, you can suppress a message in MPF, and use the MRT to display it.

The NetView SSI checks the loaded MRT definitions to designate the action to be taken. For example, the message can be suppressed, attributes of the message can be changed, or the message can be sent to NetView for automation. Attributes include color, route codes, descriptor codes, and more.

The message goes to the MRT processing before it is displayed on the system console. If you modify the message in the MRT, you modify the message that is displayed on the system console.



Important: Be careful when modifying or suppressing messages. The modified message routes to the remaining applications on the SSI bus and also the system console. This can adversely impact other applications.

REVISE command

IBM

REVISE command

```
>>--ALL-->
--REVISE-- +--+ +-----+
      +- CMD -+
      | - MSG -|
      |
>--- OFF ----->
+- STATUS-----+
+- REPORT-----+
`- MEMBER=membername +-----+
      +- REPORT-----+
      +- TEST-----+
      | .- TESTMODE=NO-. |
      | +-----+ |
      | - TESTMODE=YES -|
```

Issue REVISE commands as follows to manage the MRT and CRT, and generate reports from NetView:

- **OFF**: Disables message or command revision
- **STATUS**: Displays active MRT or CRT name and when it was loaded
- **REPORT**: Displays statistics and usage information about the active MRT or CRT
- **MEMBER**: Loads or tests syntax of an MRT or CRT

2-24

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide illustrates the syntax of the REVISE command, which replaces the old REVISMSG command that handled only MRT.

REVISE MSG, MEMBER= *membername* loads an MRT where *membername* is a set of MRT statements in DSIPARM. The MRT can contain other members with use of %INCLUDE. Data REXX is also supported.



Note: The statements load in the order given, but the SSI processes the statements in a different order than the load order.

REVISE MSG,REPORT generates a report that includes the following information:

- Number of messages that have gone through the MRT
 - Number of messages suppressed
 - MRT activity based on statement within the MRT

The REVISE command supports other parameters. See the *Command Reference* manual for more details or the online help, such as “HELP REVISE”.

Message revision features



Message revision features

- Not necessary to route messages through NetView automation table to change message text or attributes
- MRT can replace MPF table:
Might still need MPF table for user exits
- Messages can be automated even after they are deleted

2-25

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

With the MRT, you can change the color of a message. This change affects the original message to be displayed on the system console and also the copy that was sent to NetView. Without the MRT, you can use the automation table to modify the message for only the copy that goes to NetView.

In most cases, you can replace the MPF settings with MRT statements. The MRT provides functions to parse and process messages.



Note: If you call user exits from an MPF list member, you still need the MPF to call the exit. Some products might also send user exits to handle message overflows at the system console.

Extended message management



Extended message management

- CNMSTYLE definitions are used in activating SSI and MRT during initialization
- MRT can remain active without NetView
 - NetView is necessary for loading, querying, or gathering statistics
 - NetView SSI and subsystem interface router task (SSIR) are necessary if you send messages to NetView for automation
- NetView can report MRT statistics and usage information

2.26

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

CNMSTYLE definitions are provided for performing the following tasks:

- Start the NetView SSI address space when NetView initializes.
- Load a Message Revision Table (MRT).



Tip: After loading, the MRT has no dependency on NetView being active. For example, if you start NetView before JES, you can start the NetView SSI, load the MRT, and stop NetView. As your system IPL continues, your automation NetView can be started to support your automation needs.

Using the REVISE MSG,REPORT command, you can display statistics and detailed usage information. You can correlate the detailed usage information to specific statements within the MRT source in DSIPARM.

Trapping messages in the MRT

IBM

Trapping messages in the MRT

```
UPON (MSGID=...)           ! Test for a single condition
UPON (MSGID=... | MSGID= ... | PREFIX=...)      ! Test for multiple conditions
  Select
    When (JOBNAME=...)
    ...
    When (MSGID=...)
    ...
    Otherwise
    ...
  End
  Select
    When (MSGID=...)
    ...
    Otherwise
    End
  UPON (JOBNAME=...)
    select
      When (MSGID=...)
      ...
      Otherwise
      end
  UPON (OTHERMSG)
  ...

```

- Each UPON statement introduces a new section of the MRT, called an *UPON group*
- Four condition types, tested in this order:
 1. **MSGID**: ID of the message, one to twelve characters
 2. **JOBNAME**: Name of the started task, one to eight characters
 3. **PREFIX**: Three-character prefix of the message ID
 4. **OTHERMSG**: This will *always* result in a successful match, unless matched by another UPON statement.

For example, MSGID is a higher-ranking condition than JOBNAME or PREFIX

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

MRT statements fall into the two following categories:

- Statements that trap messages. This slide shows the statements that are used for trapping messages.
- Statements that take actions on trapped messages. Discussion about MRT actions occurs in a few slides.



Note: Mixed-case text is supported.

Within the MRT, the UPON statement identifies groupings of conditions that trap one or more messages. These conditions are collectively called an *UPON group*. Each UPON group defines a new section of the MRT.

UPON conditions can be one or more combinations of the following conditions:

- MSGID: Traps a specific message ID or substring of a message ID if an asterisk (*) is used.
- JOBNAMES: Traps all messages from the specified job. An asterisk (*) is supported as a wildcard.
- PREFIX: Traps all messages that match the specified three-character prefix.
- OTHERMSG: Defines default actions for messages that are not already processed (trapped) in the MRT. The OTHERMSG condition is not allowed with other UPON conditions. There can be only one UPON(OTHERMSG) condition in an MRT.

The example that this slide shows contains four UPON groups, including one UPON(OTHERMSG). The UPON conditions are ranked in the order shown. MSGID is the *highest ranking* UPON group, and UPON(OTHERMSG) is the *lowest ranking* UPON group.

Multiple conditions can be specified within an UPON group by using an OR symbol (|). The AND symbol (&) is not supported.

An UPON group ends when the next UPON group is encountered or the end of the MRT is reached. Within an UPON group, you can code one or more SELECT statements as follows for parsing the message:

- SELECT: Introduces a series of WHEN statements.
 - WHEN: Specifies one or more conditions to take action on.
- WHEN statements can test other conditions of the current message. Example conditions include console name, work queue element (WQE) bits, descriptor code, route code, and checking of syslog being on or off.
- OTHERWISE: Processes all *unmatched* conditions for a particular SELECT statement.
 - END: Identifies the end of a SELECT statement.



Note: Only the first line of a multiline message is examined. Actions taken, such as changing the color of a multiline message, apply to all lines of the multiline message.

you can code MRT comments in two ways:

- Starting the first column with an asterisk (*).
- Within MRT statements, following an exclamation point (!), as the following example shows:

```
UPON(MSGID='IEF403I') ! Test for a single condition
```

MRT search order

IBM

MRT search order

- When a message is run through the MRT, a fast-search algorithm is used for finding the relevant UPON group
- After a match is found in an UPON comparison, lower-ranking UPON conditions are not examined:
Additional SELECT statements that are coded in the current UPON group are to be processed
- The search order might not match the order of the statements that are coded in the MRT source

2-28

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

UPON statements are used for trapping messages that run through the MRT. If a match is found, subsequent (*lower ranking*) UPON conditions are not examined. You can code multiple SELECT statements to take multiple actions for the message.

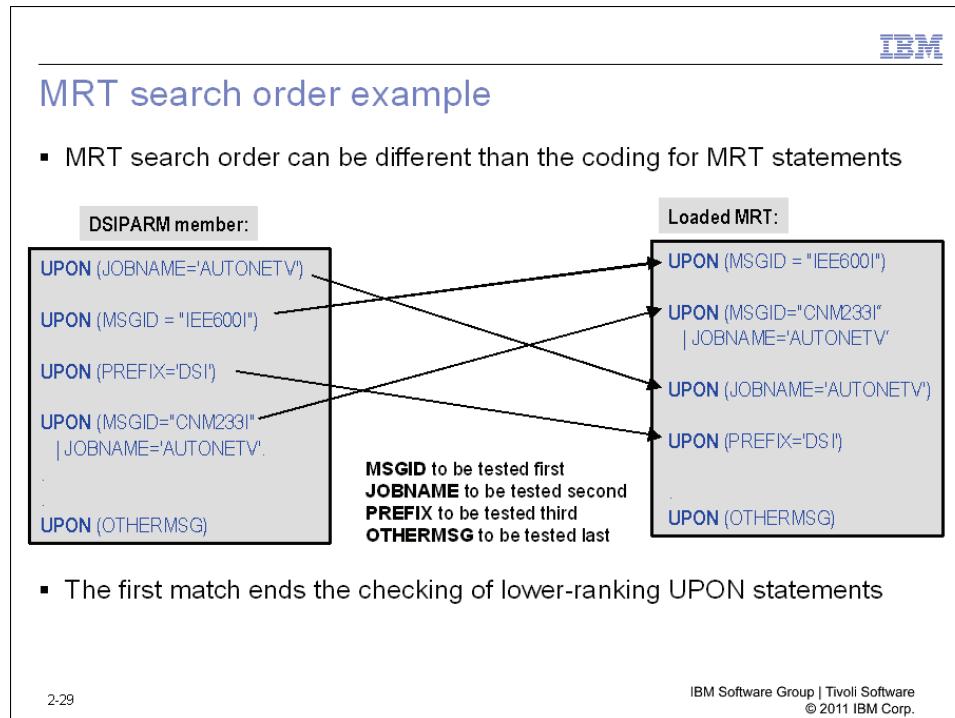
Each *UPON group* can test for a message ID, z/OS job name, or message prefix. Each of the condition types is assigned a priority with MSGID condition checks that have the highest priority and PREFIX condition checks that have the lowest priority.



Important: Because of the priority assigned to the condition types, the MRT that is loaded into storage in the NetView SSI might not match the exact order of the MRT statements as coded in DSIPARM. This can cause confusion when you attempt to debug the reason that your MRT outcome did not meet your expectations.

An UPON group that tests a message prefix (PREFIX) is a lower ranking UPON when compared to one that tests for a message ID (MSGID). UPON(OTHERMSG) is the lowest ranking UPON condition of an MRT.

MRT search order example



This slide shows an example MRT source in DSIPARM and the order of the statements in the actual MRT that is loaded in storage in the NetView SSI. Key points to note are as follows:

1. The second statement in the MRT source is loaded as the *highest ranking* statement because it contains a test for MSGID. In this example, the test is for only MSGID.
2. The fourth statement in the MRT source is loaded as the second highest ranking statement because it also contains a MSGID test. It is loaded after the test for only MSGID because the check for JOBNAME reduces its priority.
3. The first statement in the MRT source is loaded next because JOBNAME is a lower priority than MSGID but higher priority than PREFIX.
4. The third statement in the MRT source is loaded as the *lowest ranking* statement because of the check for PREFIX only.
5. UPON(OTHERMSG) is always the last statement in the MRT to load.

MRT actions

IBM

MRT actions

- EXIT: Stops any further message revision when a condition is matched:
Useful in an UPON group with multiple SELECT statements for preventing further actions from occurring
- NETVONLY: Sends message to NetView automation only
 - Suppresses display, logging, and sysplex routing of the message
 - Queues messages to NetView
- REVISE: Includes revision actions

2-30

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Three actions that are available are EXIT, NETVONLY, and REVISE. You can specify more than one action.

- EXIT: Ends with the current condition that matched. No further tests occur. This is analogous to a CONTINUE(STOP) in the automation table.
- NETVONLY: Message goes to NetView for automation only. The message is not displayed, not logged, and not routed.
- REVISE: You can modify the message text, attributes, descriptor code, route code, and more.

Using the MRT to modify messages



Using the MRT to modify messages

- REVISE action can be used for modifying the following items:
 - Color of a message
 - Attributes of a message
 - Route codes or descriptor codes
 - Can also route message to a different console
 - Destination: automation, logging, or display
 - Text of a message
 - Length restricted to 127 characters by z/OS
 - Text modifications can affect automation: Be careful
- Automation is performed by NetView automation table

2-31

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The MRT does not perform automation. Messages that run through the MRT must be passed to NetView and the automation table for automation to occur. You can use the MRT REVISE action to modify attributes of a message, such as its color, route codes, descriptor codes, and so on.

Revision examples



Revision examples

REVISE ('cr hb' COLOR)	! Change message to red, attribute to blinking
REVISE (1.* 1 "919-555-5555")	! Append phone number to end of text
REVISE ('N' AUTOMATE)	! Do not automate this message
REVISE ("Y" DELETE)	! Delete this message
REVISE ('N' DISPLAY)	! Do not show message at the console
REVISE ('N' SYSLOG)	! Do not write this message to the system log
REVISE (ROUTEZERO)	! Set all route codes to zero (false)
REVISE ("WHOKNOWS" CONSNAME)	! Send message to WHOKNOWS console

2-32

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Modifying messages with the MRT

IBM

Modifying messages with the MRT

```

UPON (MSGID = "IEE600I" | MSGID = 'IEE136I' | MSGID='IEE305I' | msgid = 'IEE457I' | MSGID = 'IEF450I' |
      MSGID='DSI803A' | msgID = 'IST453I')

  SELECT
    WHEN (MSGID = "IEE600I")           ! Notice double or single quotes (like REXX)
      REVISE ('cb hb' COLOR)          ! Change color to blue, attribute to blinking
    WHEN (MSGID = 'IST453I')           ! Change color to red
      REVISE ('CR' COLOR)
    WHEN (MSGID = 'DSI803A')           ! Change to reverse video, append "or Not!" after message
      REVISE ('HR' COLOR 1.* 1 'or Not!' NW)
      NETONLY                         ! Send to NetView only
    WHEN (MSGID LEFT 3 = 'IEE')        ! Trap a subset of messages: IEE
      REVISE ('11xx0xxx' FLGRTC1)     ! Send to route code 1 and 2, but not 5
      otherwise REVISE ('cw COLOR') EXIT ! Change message color to white and exit
  END

  SELECT
    WHEN (MSGID='IEE136I')            ! Process IEE136I
    ...
    WHEN (MSGID='IEE305I')            ! Process IEE305I
    ...
    WHEN (MSGID='IEE457I')            ! Process IEE457I
    ...
  otherwise
END

```

2-33

IBM Software Group | Tivoli Software
 © 2011 IBM Corp.

In the slide example, this UPON group traps seven messages and processes all of them in two single SELECT statements. The second SELECT statement processes the IEE messages individually.

The IEE600I message is trapped by an exact match for its message ID and is not passed to the remaining WHEN statements. If there were another SELECT statement, the IEE600I would be passed to it. It is to be displayed as blue and blinking on the system console.

Three of the messages that the UPON statement (IEE136I, IEE305I, and IEE457I) trapped match the WHEN (MSGID LEFT 3 = 'IEE') statement in the first SELECT. The messages for their route codes undergo modification. The second SELECT statement processes each individual IEE message.

Because the IEF4501 message has no WHEN statement, the OTHERWISE statement traps and processes it. The IEF4501 message is to be displayed on the system console in white.

The REVISE action is very similar to the NetView PIPE EDIT stages. For more information on other possible revision actions, see the *Programming: NetView PIPEs* manual.



Note: Multiple REVISE statements are supported or combinable into one statement.

System console example

The screenshot shows a system console window titled "System console example". The window displays a list of messages with their timestamps, source, and content. Some messages are highlighted in blue or green boxes, indicating they have been revised. The messages include various system status and error reports, such as "DSI802A reply CLOSE or MSG", "AOF568I 14:21 : STATUS OF TIVED2.AOFDAO", and "FKX701I THE CURRENT STATUS OF SP TIVED1 IS DEGRADED". Below the messages, there is a block of MRT source code. The code defines several UPON groups for different message types (e.g., DSI802A, AOF568I, FKX701I) and conditions (e.g., MSGID, PREFIX). It includes revisions for message text, colors (e.g., blue/reverse, red/blink), and automation handling.

```
*14.14.53 TIVED1      *08 DSI802A reply CLOSE or MSG
- 14.21.25 TIVED2 STC07577 AOF568I 14:21 : STATUS OF TIVED2.AOFDAO
- OUTBOUND GATEWAY TO DOMAIN
- AOFDA IS INACTIVE - OPERATOR TASK GATAOFPDB NOT DEFINED AT AOFDA
- 14.24.15 TIVED2 STC07577 AOF570I 14:24:15 : ISSUED "INGMTRAP"
- NAME=OMIIMVSB XTYPE=XREP" FOR
- MONITOR PROCESSING OF XREPMONB
- 14.25.09 TIVED1 STC08226 DSI208I TIME EXPIRATION - ID= 'FKX00001' - CMD=
- 'FKXEACT2 TIVED1 TIVED1 SP'
- 14.25.48 TIVED2 STC07586 EC242: CT/DS RECONNECT NOT SUCCESSFUL
- 14.29.15 TIVED2 STC07577 AOF570I 14:29:15 : ISSUED "INGMTRAP"
- NAME=OMIIMVSB XTYPE=XREP" FOR
- MONITOR PROCESSING OF XREPMONB
- 14.29.50 TIVED1 STC08226 FKX701I THE CURRENT STATUS OF SP TIVED1 IS
- DEGRADED

UPON (MSGID = 'DSI802A')           ! change text of these messages
| MsgID = 'DSI803A')
- revise(wl 1 msgid nw          ! put in reply ID and msgid
"reply CLOSE or MSG" nw) ! Change message text too
UPON (PREFIX = 'AOF')              ! SA z/OS messages
REVISE ('CB HR' COLOR)            ! make messages blue/reverse
REVISE ('N' AUTOMATE)             ! do not pass to automation
UPON (MSGID = "AOF568I")          ! SA z/OS message : Gateway Error
REVISE ('CR HB' COLOR)            ! make messages red/blinkling
REVISE ('N' AUTOMATE)             ! do not pass to automation
UPON (MSGID = "FKX701I")          ! AOH/TCP Stack degraded message
REVISE ('CG HR' COLOR)            ! make messages green/reverse
```

2-34

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This example shows the MRT source from DSIPARM and the results on the system console. The MRT source contains four UPON groups. Several of the messages shown here have been revised with the MRT.

The text of the DSI802A message was altered completely. Without message revision, the DSI802A would look similar to the following text:

DSI802A *domainid* REPLY WITH VALID NCCF SYSTEM OPERATOR COMMAND



Important: Be careful when modifying the text of messages. The modified message text can be passed to automation. The automation might not function properly because of the text of the message.

The AOF568I message is red, yet an UPON group is coded for all AOF messages (PREFIX=AOF) to change the messages to reverse video blue. The UPON group for AOF568I uses a MSGID condition, but the UPON group for AOF messages uses the PREFIX condition. Because the MSGID condition is a *higher ranking* condition, it is tested and processed before the PREFIX condition.

Revision variables



Revision variables

- Command SETRVAR and sample CNMSRVAR can be used for setting revision variables
- Revision variables can be resolved by MRT and CRT order, RVAR
- Allows to change the behavior of the revision tables without loading a different table
- CNMSMRT1 example shows CHRON can be used for setting variables that indicate shift or holiday

```
SELECT
WHEN (W2 != /STATCOMP) ! starting special proc? see next WHEN
WHEN ("SHIFT" RVAR = "NORMAL") !
NETVONLY=CNMSRVMC ! double use of sample clist!
WHEN ("SHIFT" RVAR = "HOLIDAY")
! cmd is allowed, no action here
```

2-35

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Student exercise

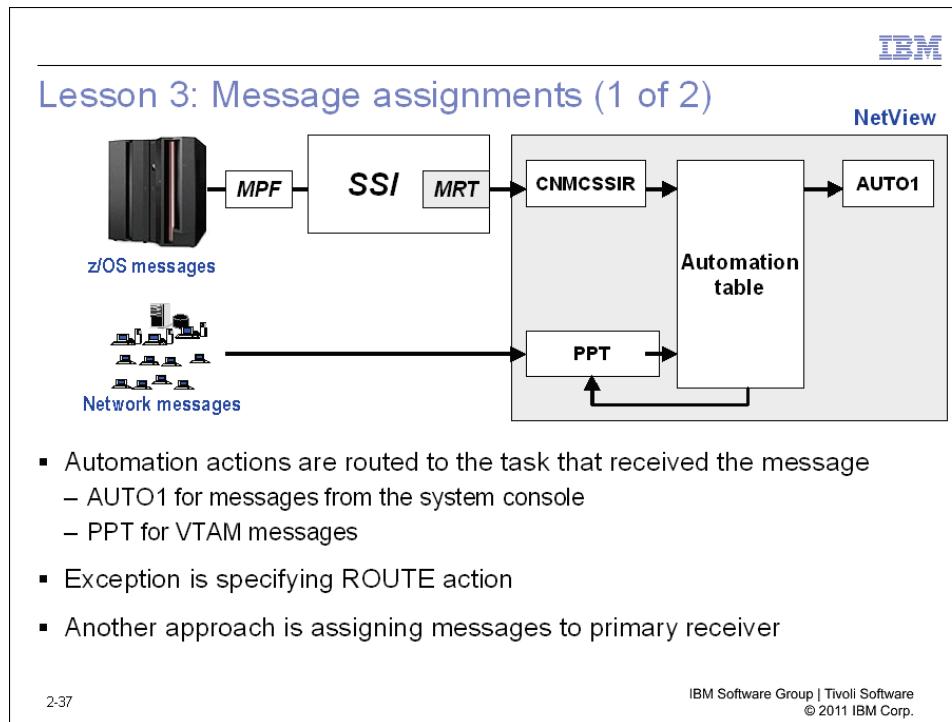
Student exercise



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 2-1.

Lesson 3: Message assignments



Messages that are matched in the automation table can result in commands being issued to take corrective actions. By default, the commands run on the task that receives the message. AUTO1 receives z/OS messages (because AUTO1 is the task that started CNMCSSIR) and the PPT task receives VTAM network messages. This approach can cause several problems:

- The task might not be active.
- The task might be active but busy, potentially creating a bottleneck.
- Not all tasks support all NetView commands. For example, the PPT task does not support commands that cause the task to wait.
- And so on

You can direct the action (and message) to one or more tasks with the ROUTE specification in the automation table. Or you can assign the message to one or more tasks. Tasks that are assigned a message are called *primary receivers* of the message.

This lesson discusses assigning messages to NetView tasks. Customers can use a combination of the ASSIGN command and automation table ROUTE specification to control where to route messages and the actions to take. Discussion of the automation table ROUTE specification comes at the time of more detailed discussion of the automation table.

Message assignments (2 of 2)



Message assignments (2 of 2)

- By assigning messages to operators, you control where automated actions take place, independent of automation table routing
- You can assign messages to individual operators or groups of operators:
 - Operator tasks (OSTs)
Be careful: Operators might log off
 - Automated operators (AOSTs)

2-38

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The ASSIGN command is used for assigning messages to NetView operators and automation operators. With the ASSIGN command, you can define a default task for receiving the message (or group of messages). The primary receiver can be the default task for automation table actions, increasing the likelihood that the action routes to an active task and runs.

Types of messages

IBM

Types of messages

Messages can be the following types:

- Solicited: Result of an operator command
 - Could result from active monitoring command
 - Most automation does not act on these commands
- Unsolicited: Unexpected, such as a resource failure message
 - Origin might be z/OS, VTAM, NetView, or other applications
 - Source can be from passive monitoring (reactive automation)
- HDRMTYPE(): Usable to test with

2-39

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

NetView distinguishes two basic types of message as follows:

- Solicited message

A solicited message is a response to a command. Solicited messages route to the task that issues the command.

- Unsolicited message

- An unsolicited message has no known destination. An unsolicited message typically indicates a problem. Examples might be a job start, a line problem, a transaction abend, or security violation.
 - An unsolicited message needs to be routed for review and action.
 - An unsolicited message is the source for reactive automation.

Unsolicited messages might come from z/OS while others come from VTAM and NetView itself. Be aware that unsolicited messages from z/OS might require different treatment than those that arrive from VTAM or NetView.

For a complete list of all possible HDRMTYPE values, see “Appendix G: NetView Message Type (HDRMTYPE) Descriptions”, in the *NetView for z/OS Automation Guide*.

Because NetView uses the concept of a known destination, some messages that seem unsolicited are treated as solicited. For example, if one NetView operator sends a message to another NetView operator, the message is treated as solicited because the destination is known.

Similarly, if a NetView user specifies the MVS VARY CONSOLE(*),ROUTE=1 command to receive unsolicited MVS messages of route code one, the treatment is different. NetView treats replies to this user as solicited messages.

Primary receiver

Primary receiver

- The primary receiver is a task defined for receiving unsolicited messages
- You can define several primary receivers for items as follows:
 - All VTAM messages
 - All z/OS messages
 - Only IEF* messages
 - Specific message ID
- The default primary receiver is PPT or SSIR task:
 - PPT task for VTAM messages
 - SSIR task for z/OS messages
- You use the ASSIGN command to define NetView (automation) tasks as primary receiver of messages or groups of messages

IBM

2-40

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

NetView uses the concept of a *primary receiver* to receive one or more unsolicited messages. You can define more than one primary receiver.

By default, the PPT task is the primary receiver of unsolicited VTAM messages, and the SSIR task is the primary receiver for unsolicited z/OS messages. The PPT task causes an error if EXECs try to wait for a response to a command.

The ASSIGN command enables the definition of automation tasks as primary receivers to ensure that automated actions occur. You can also use the ASSIGN command for defining groups of operators as primary receivers. Because unsolicited messages are used for reactive automation, you define automation tasks as primary receivers.

Operator groups



Operator groups

- Reasons to define groups of operators
 - For increasing likelihood that a message routes successfully
 - So that an operator can be in more than one group
 - Modify the operator initial clist (IC) to issue ASSIGN commands when the operator logs on
- Operator group examples
 - +SHIFT1: All first-shift operators
 - +CICSOPS: Operators who are to receive CICS messages
 - +ABENDOPS: Specialized group of operators who are to receive abnormal end (ABEND) messages

2-41

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can route automation actions to a group of operators. The operators in the group might change, but you do not need to change the automation table. For example, you can define a group of operators to process all CICS events. As NetView operators log on, they can be dynamically added to the +CICSOPS group. When their shift changes and they log off, they can be dynamically deleted from the group.

ASSIGN and LIST commands



ASSIGN and LIST commands

- Use the ASSIGN command to perform tasks as follows:
 - Define operators that are to receive copies of solicited and unsolicited messages
 - Operators defined to receive unsolicited messages, who are the primary receiver and become the default task for actions from the automation table
 - An automation operator (AOST) who is always logged on to NetView
 - If not logged on, the action might not route
 - Define groups of operators
- Use the LIST command to display message and operator group assignments

2-42

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can use the NetView ASSIGN command to define primary receivers, operators who should receive copies of messages, and groups of operators.

ASSIGN MSG examples



ASSIGN MSG examples

ASSIGN MSG=IST530I,PRI=(NETOP1,NETOP2),SEC=(NETOP3,OPER1)

- Define primary receiver for IST530I message:
 - Route all unsolicited IST530I (multiline) messages to NETOP1
 - If NETOP1 not logged on, forward messages to NETOP2
- If primary routing occurs, secondary copies route to both NETOP3 and OPER1:
 - You add HDRMTYPE=* to send copied message
 - Copied message is not eligible for automation

ASSIGN MSG=IEF*,PRI=(AUTO1,AUTO2)

- Define primary receiver for all IEF unsolicited messages
- Route to AUTO1 or AUTO2 if AUTO1 is not active

2-43

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Use ASSIGN MSG to define primary receivers (PRI), secondary receivers (SEC), and also operators who should receive a COPY of the message.

The first example explains how to use PRI and SEC. NETOP1 and NETOP2 are defined as the primary receiver for IST530I. If NETOP1 is not logged on, NETOP2 receives the IST530I message. Secondary copies of the message route to NETOP3 and OPER1. If the IST530I does not route to a primary receiver, the secondary routing does not occur.

The second example defines AUTO1 and AUTO2 as the primary receiver for all IEF messages. The receivers are as follows:

- **PRI:** Defines a list of operators to receive unsolicited messages. (Defines the primary receiver for the message.)
 - Message routes to the first logged-on operator in the list.
 - Message is to be displayed with a percent sign (%) to identify the primary receiver.
- **SEC:** Defines a list of operators to receive copies of unsolicited messages.
 - Message routes to all logged-on operators if it was sent to the primary receiver.
 - Message is not automated.
 - Message is to be displayed with an asterisk (*) to identify it is a copy of an authorized message.
- **COPY:** Defines a list of operators to receive copies of solicited messages.
 - Message is not automated.
 - Message is to be displayed with a plus sign (+) to identify it is a copy.

ASSIGN GROUP examples



ASSIGN GROUP examples

ASSIGN MSG=IEC501A,PRI=(MVSOP),SEC=(+TAPEOPS)

Assign operator MVSOP as the primary receiver for message IEC501A and the +TAPEOPS group as secondary receivers

ASSIGN GROUP=+TAPEOPS,OP=(TAPEOP1,TAPEOP2)

Define the +TAPEOPS group with two operators

ASSIGN GROUP=+TAPEOPS,OP=TAPEMGR,ADDLAST

Dynamically add TAPEMGR to group definition upon logon

ASSIGN GROUP=+TAPEOPS,OP=TAPEMGR,DELETE

Dynamically delete TAPEMGR from group definition upon logoff

2-44

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Use ASSIGN GROUP to define operator groups. You can add operators to a group dynamically. The slide examples show how to route a message (IEC501A) to MVSOP as the primary receiver and an operator group, +TAPEOPS, as secondary receivers. The operator group can be defined before or after the ASSIGN MSG is issued.

Initially, +TAPEOPS is defined with two operators: TAPEOP1 and TAPEOP2. When the manager, TAPEMGR, logs on, it is dynamically added to the +TAPEOPS group as the last task in the list. When TAPEMGR logs off, it is dynamically removed from the +TAPEOPS group.

Displaying the primary receiver



Displaying the primary receivers

- Issue the **LIST ASSIGN** command to display the primary receivers for IEC501A message:

```
LIST ASSIGN=AUTH,MSGID=IEC501A
DSI636I AUTH MESSAGE STRING: 'IEC501A'
BNH647I PRIORITY LEVEL: 3
DSI638I PRI(1ST): MVSOP
DSI639I SEC(ALL): +TAPEOPS
DSI642I END OF ASSIGN DISPLAY
```

- To display the primary receiver for all messages, issue either of the following commands:
 - LIST MSG=AUTH
 - LIST ASSIGN=AUTH,MSGID=ALL

2-45

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide example displays the message assignment for the IEC510A message that the previous slide used. The +TAPEOPS operator group is listed, but none of the operators.

Displaying ASSIGN GROUP definitions



Displaying ASSIGN GROUP definitions

Issue the LIST ASSIGN command to display the operator groups that you have defined:

```
LIST ASSIGN=GROUP,GROUP=+TAPEOPS
DSI180I GROUP ID: +TAPEOPS
BNH647I  PRIORITY LEVEL: 3
DSI640I  OP(ALL): TAPEOP1 TAPEOP2 TAPEMGR
DSI642I END OF ASSIGN DISPLAY
```

2-46

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide example shows how to display the members of the +TAPEOPS operator group.

Student exercise

IBM

Student exercise



247

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercises 2-2 and 2-3.

Summary



Summary

Now that you have completed this unit, you can perform the following tasks:

- Describe the functions provided by the message processing facility (MPF)
- Describe the functions provided by the message revision table (MRT)
- Assign a primary receiver for unsolicited messages

Unit 3: NetView commands for facilitating automation

IBM

Unit 3:

**NetView commands for facilitating
automation**



© 2011 IBM Corp.

Introduction

This unit discusses the NetView commands that you can use for facilitating automation. Commands include those for establishing communication to other NetView domains and other applications, and commands for scheduling and managing timers. Using timers is very important when monitoring resources.

Objectives



Objectives

When you complete this unit, you can perform the following tasks:

- Implement the NetView-to-NetView communication required for multisystem automation
- Route commands to NetView operators and automation tasks
- Use NetView timers to proactively monitor resources

Lesson 1: NetView-to-NetView communication



Lesson 1: NetView-to-NetView communication

You can configure each NetView domain communicate with other domains

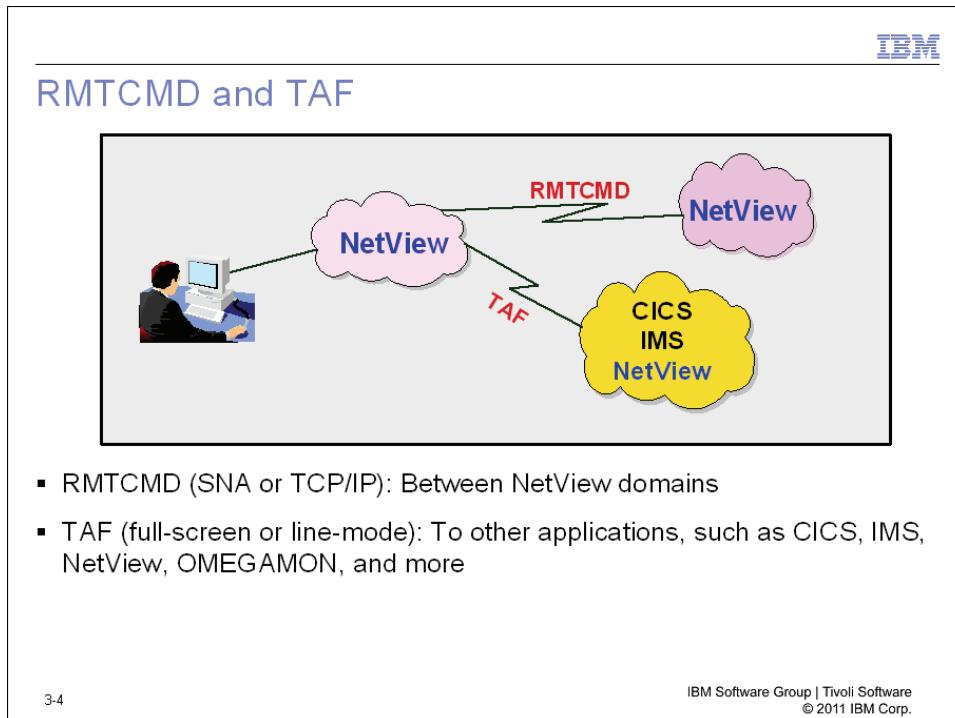
- RMTCMD over SNA (LU 6.2) or TCP/IP
Called cross-domain communication
- Terminal Access Facility (TAF) full-screen
- Focal-point architecture that connects NetView domains

3-3

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This lesson focuses on communication between NetView domains, commonly called *cross-domain* communication, and other applications. These connections are key in implementing multisystem automation. This lesson also includes discussion about RMTCMD, TAF, and focal point architecture discussions.

RMTCMD and TAF

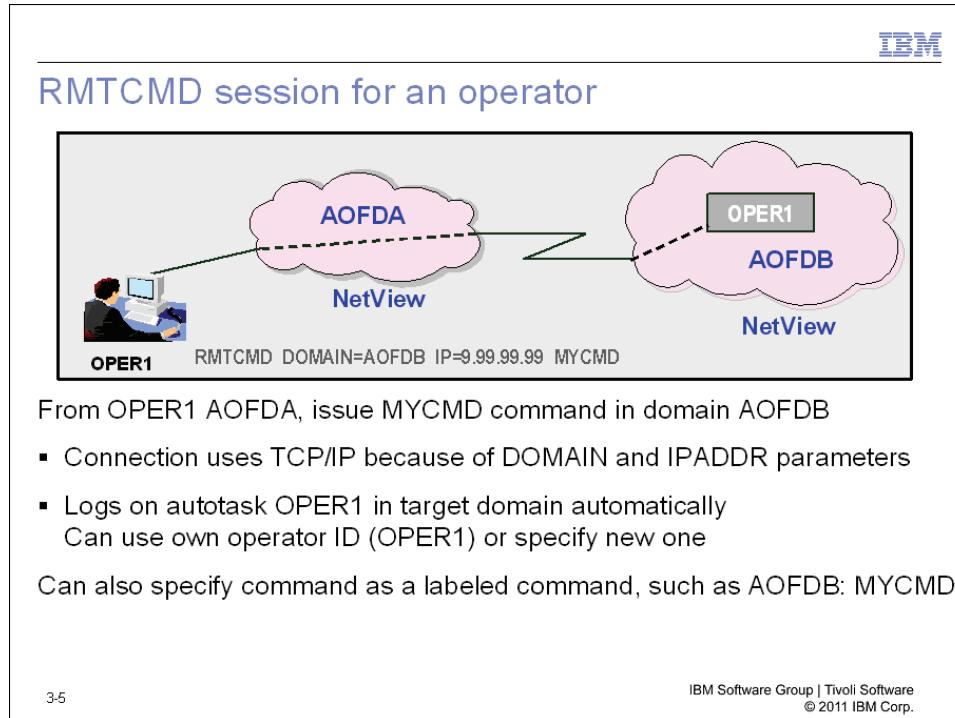


RMTCMD and TAF connections to another system provide the basis for multisystem automation. RMTCMD establishes connections to remote NetView domains. You can use TAF sessions for establishing connections to applications, such as IMS, CICS, and OMEGAMON.

TAF supports both full-screen and line-mode sessions. You can use line-mode sessions for receiving messages and issuing commands directly to the target application. *Screen scraping* automation uses full-screen sessions for displaying panels on VOSTs and interrogate fields on the panels. The TAF commands are as follows:

- BGNSESS: Start a session (full-screen or line-mode).
- SENDSESS: Route commands over a line-mode session.
- LISTSESS: Display the active TAF sessions.

RMTCMD session for an operator



The RMTCMD command sends commands to another NetView domain by using either a SNA LU6.2 or TCP/IP session. Both IPv4 and IPv6 are supported.

If a specific operator task is specified and the task is active, that task is used for the command. (The operator task might be logged on at the remote domain, or the task might be an autotask.) If the operator is not active in the target domain, it is logged on as an autotask before the command runs. Whenever a RMTCMD command is issued to a new domain and autotask combination, a remote session begins for that user.

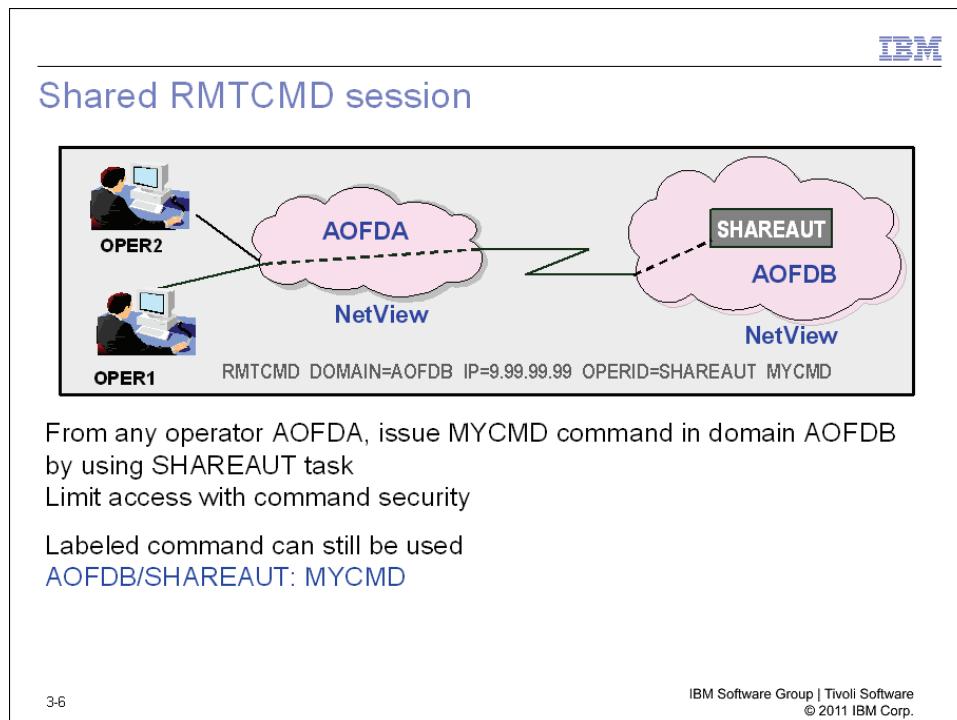
If a RMTCMD is issued to a NetView domain but no operator ID is specified, an existing session is used. If the caller does not have an existing session, a new session starts, using the operator ID of the caller.



Note: The caller must be defined in the DSIOPF member at the target NetView.

Labeled commands are an abbreviated format for issuing RMTCMD commands. To route the MYCMRD command to NetView AOFDB domain, issue the AOFDB: MYCMRD command. You use an existing session or start a new session by using the operator ID of the caller.

Shared RMTCMD session



Several users from several source domains might route requests to a single domain and autotask combination. Such requests queue and run serially at the target domain and might not run immediately. Or the requests might not run at all. Results from such executions return to the caller as a single block of messages when the command has finished.

In the example, correlated responses return to the originating operator (OPER1 or OPER2). All operators at source systems (for example, AOFDA domain) can use the SHAREAUT autotask for their RMTCMD session to the AOFDB domain.

Labeled commands are supported. To run the MYCMD command on the SHAREAUT task in the NetView AOFDB domain, issue the AOFDB/SHAREAUT: MYCMD command.

CNMSTYLE definitions



CNMSTYLE definitions

- CNMSTYLE usable for defining defaults for the rmtsyndef parameters as follows:

RMTINIT.IP = YES

RMTSYN.netID.domain =

(TCP/IP) *IP_host_name | IP_address / port_num*

(LU 6.2) **SNA**

- Not necessary to specify IP and PORT parameters on RMTCMD

3-7

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

CNMSTYLE contains two statements for defining RMTCMD communication as follows:

- The RMTINIT statement enables IP or SNA (not shown on slide) over RMTCMD.
- The RMTSYN statement defines RMTCMD connectivity (SNA LU 6.2 or TCP/IP) between NetView domains.

The RMTSYN statement simplifies use of RMTCMD over TCP/IP. You do not need to specify an IP address on your RMTCMD command if it is defined on a RMTSYN statement. For example, RMTCMD DOMAIN=*domain,command* uses the IP address and port parameters from a RMTSYN statement.



Note: RMTCMD requires NetView tasks as follows:

- DSHPDST and DSUDST for SNA LU 6.2
- DSITCPIP for TCP/IP communication

RMTSYN definition examples



RMTSYN definition examples

- As an example, define RMTCMD over TCP/IP to AOFDA and AOFDB domains. Use the default port number (4022) and RMTCMD over SNA to the AOFDC domain as follows:

NETID = NETA

RMTINIT.IP = YES

RMTSYN.NETA.**AOFDA** = 10.44.15.200

RMTSYN.NETA.**AOFDB** = 10.44.15.201

RMTINIT.SNA = YES

RMTSYN.NETA.**AOFDB** = SNA

RMTSYN.NETA.**AOFDC** = SNA

3-8

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide shows examples of using RMTSYN to define RMTCMD sessions:

- RMTCMD over TCP/IP is enabled. The target IP address for two domains (AOFDA and AOFDB) is defined.



Note: You can start RMTCMD sessions to other domains that are not in the RMTSYN definitions. RMTSYN is not required. If you do not define a RMTSYN statement, you must specify all parameters on your RMTCMD when starting the session.

- RMTCMD over SNA LU 6.2 is enabled to two domains (AOFDB and AOFDC).

To start a RMTCMD session to domain AOFDB by using the same operator ID, enter **RMTCMD DOMAIN=AOFDB,command**.

If not specified, RMTCMD over TCP/IP connections use the port that is defined for the DSIUDST NetView task (default of 4022). An example follows:

```
LIST DSUUDST
TYPE: OPT TASKID: DSUUDST TASKNAME: DSUUDST STATUS: ACTIVE
MEMBER: DSUINIT SECURITY: NONE AUTH: SENDER
HOSTNAME=TIVED1 HOSTID=10.44.15.200
PORT=4022
SOCKETTYPE=AF_INET
LOADMOD: DSIZDST
Task Serial: 721
Messages Pending: 0 Held: 0
WLM Service Class: Not Available
END OF STATUS DISPLAY
```

See the *Administration Reference* manual for more details.

Query RMTCMD sessions: Outbound

IBM

Query RMTCMD sessions: Outbound

RMTCMD QUERY RMTDOMS

```
BNH060I RMTCMD QUERY INFORMATION
BNH061I -----
BNH068I REMOTE NETVIEW      VERSION   TRANSPORT
BNH061I -----
BNH069I ADCD.AOFDB          V5R4      SNA
```

Display the domains you have started a RMTCMD session to

RMTCMD QUERY RMTAUTOS LU=AOFDB

```
BNH072I RMTCMD QUERY INFORMATION FROM ADCD.AOFDB
BNH061I -----
BNH070I NETOP1
```

Display the autotask in the target domain

Note: These commands were executed in AOFDA domain

3-9

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

RMTCMD QUERY displays session data for both SNA LU 6.2 and TCP/IP sessions.

Query RMTCMD sessions: Inbound

IBM

Query RMTCMD sessions: Inbound

RMTCMD QUERY LCLAUTOS

```
BNH060I RMTCMD QUERY INFORMATION
```

BNH061I	-----						
BNH064I	RMTCMD	ORIGIN	ORIGIN	ORIGIN	EXP		
BNH065I	AUTOTASK	NETVIEW	OPERATOR	VERSION	VALUE		TRANSPORT
BNH061I	-----		-----	-----	-----	-----	-----
BNH066I	NETOP1	ADCD.AOFDA	NETOP1	V5R4	N/A	SNA	
BNH066I	OPER1	ADCD.AOFDA	NETOP1	V5R4	N/A	SNA	

NETOP1 in AOFDA domain started a RMTCMD session to *this domain* (AOFDB)

Note: This command was executed in AOFDB domain

The slide shows two RMT sessions, both started by NETOP1 in AOFDA. The first one uses NETOP1 as user in AOFDB, while the second one uses OPER1.

RMTCMD DOMAIN=AOFDB,OPERID=XXXX,MYCMD

If you do not specify an OPERID, the operator ID of the caller is used.

Lesson 2: Routing commands

IBM

Lesson 2: Routing commands

- Commands can be run on other NetView tasks
 - EXCMD
Originating task receives no output from the command execution
 - Labeled command
Output returns to the task that issues the labeled command
- Example is as follows:
 1. Primary receiver (SYSAUTO) is assigned many system messages
 2. To avoid a backlog on the primary receiver, routing the actions to other related autotasks (SYSAUTO1 through SYSAUTO9) occurs

The diagram illustrates the routing of automated actions. On the left, a box labeled "Automation table" contains a graphic of five horizontal lines of varying lengths. An arrow points from this box to a central box labeled "SYSAUTO". From the "SYSAUTO" box, three arrows branch out to three separate boxes labeled "SYSAUTO1", "SYSAUTO9", and "SYSAUTOx". Below the "SYSAUTO" box, the text "EXCMD SYSAUTOn" is written.

- 3. SYSAUTO routes automated action to one of the SYSAUTOn autotasks

3-11

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

NetView provides the EXCMD command, usable to route a command to another task. The specified command runs if the task is authorized.

The slide shows an example of EXCMD use. Suppose you have a SYSAUTO automation task that is defined as the primary receiver of a group of messages. The messages do not have to be related. Because SYSAUTO is assigned to receive the messages, the actions that are defined in the automation table run under SYSAUTO, by default. You can use SYSAUTO to act a *router task* by routing the actions to a pool of automation tasks: SYSAUTO1 through SYSAUTO9 in this case.

This approach is usable for reasons as follows:

- The logic in the automation table is simplified to route the messages and actions to the default automation task, SYSAUTO.
 - SYSAUTO checks for the next available SYSAUT0xx automation task and routes the actions. Distributing the automation across multiple automation tasks affects overall throughput.
 - By separating the routing function from the actions being routed, you can avoid congestions on the automation tasks that perform the automation.

Example of routing commands



Examples of routing commands

- EXCMD:
 - Route the LOGOFF command to AUTO1:
EXCMD AUTO1,LOGOFF
 - The task that issues the EXCMD does not receive a response
- Labeled command:
 - Route a LIST STATUS=TASKS command to AUTO1:
/AUTO1: LIST STATUS=TASKS
 - The originating task receives results

Security consideration: Only authorized operators should be authorized to EXCMD a LOGOFF command

3-12

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

As an example, OPER1 issues the following command:

EXCMD AUTO1,LOGOFF

This command causes the AUTO1 task to log itself off by running the LOGOFF command. This example also points out that the use of EXCMD can be a security issue. You should control the tasks that have access to issue an EXCMD and also the commands that route with EXCMD. You can also use AUTHCHK=SOURCEID to have the command checked under the origin task. See the *NetView Security Reference* manual for more details on command security.

When EXCMD is used, no output from the routed command returns to the task that routes the request (OPER1, in this example). You can use *labeled commands* to see the output.

As an example, OPER1 issues the following labeled command:

/AUTO1: LIST STATUS=TASKS

The LIST command runs on the AUTO1 task, but the command responses are displayed to OPER1.

Student exercise

Student exercise



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 3-2.

Lesson 3: Timer commands



Lesson 3: Timer commands

NetView commands for managing timers

- AT: Schedule a command to run once, at a specific time
- EVERY: Schedule a command to run periodically by using a timer interval that is provided
- AFTER: Schedule a command to run once, relative to the current time
- CHRON: Use interface to handle AT, EVERY, AFTER, and other commands
- LIST TIMER: Use command interface to display timers
- PURGE TIMER: Use command interface to delete timers
- TIMER: Use panel interface to manage timers (display, modify, and delete)

3-14

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This lesson focuses on NetView commands for scheduling actions, commonly referred to as *timer commands*. They are the basis for proactive automation. You use timers to schedule commands for collecting resource information.

Scheduling a timer



Scheduling a timer

1. Assign unique timer ID
 - You specify ID=keyword
 - You specify one to eight characters
 - NetView default is SYSnnnnn
2. Route to a task
 - You specify ROUTE=keyword
 - You can specify any task
 - Task can be an issuing task (default), PPT task, or operator group
 - Task must be active when timer expires
3. Enter time format
 - You use ddd hh:mm:ss convention
 - You specify days, hours, minutes, and seconds
 - DEFAULTS and OVERRIDE settings control time format
4. Save timer to the Save/Restore database

3-15

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Each timer has an associated timer ID that can be up to eight characters long. The timer ID must be unique. If you do not specify a timer ID, NetView generates one that uses a prefix of SYS. Operators cannot specify a timer ID that uses SYS for a prefix.

You can schedule timers on the task that issues the timer command or on another NetView task by specifying the ROUTE operand. The task does not need to be active when you schedule the timer (AT, EVERY, AFTER, CHRON). However, the task must be active to run the command when the timer expires. Security can be performed on the origin or target task.

Timers can be saved to the NetView Save/Restore database (using task DSISVRT) and restored when NetView re-initializes. This action prevents having to set the timer every time NetView starts.

Supported commands



Supported commands

The command can be any as follows:

- Valid NetView command
- Valid NetView application command
 - For example, SA z/OS commands
- User-written procedure
 - REXX
 - Command list
 - Assembler
 - PL/1
 - C

3-16

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can schedule timers to issue any valid NetView command, including commands from products such as SA z/OS and user-written commands.

Routing commands to the PPT



Routing commands to the PPT

- Can use PPT task because it is always active
- Some considerations
 - Performance might not be optimal:
Better to route to an automation task than consume cycles on the PPT task
 - PPT does not support all functions:
For example, WAIT is not allowed

3-17

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can schedule timer commands to run under the NetView PPT task. Limit the amount of activity on the PPT task because the task is vital for several NetView functions.

Timers for active monitoring



Timers for active monitoring

Use active monitoring timers to schedule commands or EXECs to poll information:

- AT
- EVERY
- AFTER
- CHRON



3-18

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Active monitoring, also known as proactive monitoring, uses NetView timers to schedule the polling of resource information. Characteristics of polling timers are as follows:

- The EVERY timer is the most common timer for active monitoring. In some cases where it might take a long time to retrieve and process the polling information, an AFTER timer is also used.
- The AFTER timer is scheduled after each polling interval completes.
- An AT timer is used for scheduling a command to run once daily. An example is performing a database backup and archive at midnight.
- A CHRON timer provides the same functions as AT, EVERY, and AFTER, but with additional features.

AT examples



AT examples

AT 12/31 23:59:59 PPT MSG ALL,Happy New Year

At midnight on December 31, use the PPT task to send a Happy New Year message to all active NetView operators

AT 11:00 ID=FRED MYCLIST

At 11:00 a.m. of the current day, issue the MYCLIST command under the issuing task, assigning a timer ID of FRED to this timer

3-19

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

AT timers run a command one time at a specific time. Examples are as follows:

- At midnight
- At 6 p.m.

EVERY examples



EVERY examples

EVERY 1:30, ID=CHKST,CHKCLIST JOB

Every 90 minutes, issue the CHKCLIST JOB command on the default (issuing) task and assign it a user ID of CHKST

EVERY 7 00:00:00 ID=WEEKLY ROUTE=AUTODB BKUPDBS

- Every seven days, issue the BKUPDBS command under the AUTODB automation task, and assign it a timer ID of WEEKLY
- Note: The time of day is 00:00:00, indicating that the timer is scheduled every seven days from the current time

3-20

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

EVERY timers run a command more than one time, using the time interval provided until the timer is purged or NetView shuts down. Examples are as follows:

- Every 5 minutes
- Every 24 hours

AFTER example



AFTER example

```
AFTER 50,ROUTE=NETOP1,  
    PIPE NETV LIST STATUS=OPS  
        LOCATE /AUTO1/ |  
        CONSOLE ONLY
```

After 50 minutes, issue the specified PIPE NETV command under NETOP1, displaying the results to the NetView console. Use a default timer ID that NetView generates (SYSnnnnn)

3-21

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

AFTER timers run a command one time relative to the current time. Examples are as follows:

- After five seconds
- After ten minutes

ROUTE and TIMEFMSG operands



ROUTE and TIMEFMSG operands

- ROUTE: Identifies target operator where the command is to run:
 - If no operator is logged on, the command does not run
 - Asterisk (*) is the issuing operator
 - Group names are supported
- TIMEFMSG: Indicates if a BNH357E error message should be issued if a timed command could not be queued to target operator:
 - Default value is NO
 - DEFAULTS and OVERRIDE settings also control TIMEFMSG

3-22

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The ROUTE parameter of a timer command (AT, AFTER, EVERY) specifies the operator task where the timed command is to run. For example, the following command runs on the OPER1 task (if OPER1 is logged on at the specified time):

```
AT 12/31 23:59:59,ROUTE=OPER1,MSG ALL,'Happy New Year'
```

A single operator or a group of operators can be specified. ROUTE=* indicates the issuing operator and is the default. A group operand indicates that the first logged-on operator in the list of group members runs the command. If no valid operator is logged on, the command is not issued.

Security can be performed on the origin or target task.

If the command is unable to be queued to the target task, specify the TIMEFMSG parameter to display a BNH357E message that the timer command failed to start. Details are in the message to identify why the command failed.

```
BNH357E  TIMER COMMAND FAILED TO START. TYPE: type ID: id TASK:  
task MQSRC: mqsric INTERVAL: int1 int2 CONTINUE: cont CMD: cmd
```

EVERYCON operand



EVERYCON operand

- EVERYCON: Indicates if timed command continues to be queued even after queuing failure occurs:
Default value is NO
- Only EVERY command uses EVERYCON
- DEFAULTS and OVERRIDE settings also control EVERYCON

3-23

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

If the timed command fails for an EVERY timer, specify the EVERYCON parameter to continue scheduling the EVERY timer (EVERYCON=YES). An example follows:

```
EVERY 1:30,EVERYCON=YES, ID=CHKST,CHKCLIST JOBS
```

The default value for the EVERYCON parameter is defined in CNMSTYLE as follows:

```
DEFAULTS.EVERYCON = yes
```

You can change the default value with the DEFAULTS and OVERRIDE commands.

CHRON example



CHRON example

```
CHRON AT=XXXX-12-31-00.00.00.000000
EVERY=(INTERVAL=(01.00.00.000000)
REMAFTER=001-00.00.00.000000)
COMMAND='MSG ALL,HAPPY NEW YEAR'
```

Every year, send a message, HAPPY NEW YEAR, to all operators every hour on December 31. REMAFTER removes the timer one day after the AT time (midnight)

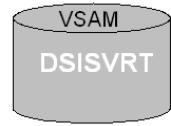
3-24

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can use the CHRON command to schedule more complex timers. In this example, December 31 is a day that is defined in the DSISCHED database:

```
XXXX-12-31 NEW_YEARS_EVE,HOLIDAY
```

Saving and restoring timers

Saving and restoring timers

- You can save timers to the save/restore database by using SAVE parameter on AT, EVERY, AFTER, or CHRON. An example is as follows:
AT 0, ID=MIDNIGHT,SAVE,SODCLIST
- You can restore each time that NetView initializes by using the RESTORE TIMER command
- You can restore all saved timers unless the time has expired
- SYSnnnn timer IDs change to RSTnnnn
- Save/restore database is also usable for global variables:
 - Task and common global conditions
 - Means for saving information across recycle of NetView, such as resource status information

3-25

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

When NetView ends, all timers that were created by earlier AT, EVERY, and AFTER commands are lost unless they were saved to the Save/Restore database. (This database is a VSAM file.) When NetView restarts, any saved timers can be restored, removing the need to re-issue the original commands.

To save a timer, use the SAVE operand on the timer command, which must precede the command to be issued. An example follows:

```
AT 0, ID=MIDNIGHT, SAVE, SODCLIST
```

Use a meaningful timer ID for timers to be saved.



Note: The Save/Restore database is also used for NetView common and task global variables.

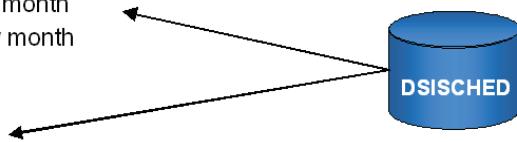
The RESTORE TIMER command retrieves all timers in the Save/Restore database. Timers are restored when NetView initializes with the INIT.TIMER=Yes statement in CNMSTYLE. If a timer has expired, it is not restored, but a message is provided to indicate the expiration.

Miscellaneous timer topics

IBM

Miscellaneous timer topics

- Combine timer commands as in following examples:
 - EVERY 7 00:00:00 ID=WKLYMID,AT 23:59:59 ROUTE=AUTODB BKUPDBS
 - Every seventh day at midnight, issue the BKUPDBS procedure on the AUTODB automation task
- Use CHRON to schedule timers as follows:
 - Every first Monday of a month
 - The last Friday of every month
 - On Christmas day
 - On Boxing Day
 - On my 50th birthday



3-26

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can combine NetView timers. This slide shows an example of combining an EVERY timer with an AT timer. In this case, the BKUPDBS command runs on the AUTODB task every seventh day at midnight, an example of a timer scheduled on a weekly interval.

CHRON timers support more functionality than AT, EVERY, and AFTER timers by using a calendar function (DSISCHED member) defined by the user. You can schedule a CHRON timer to run on a team member's birthday.

Displaying timers



Displaying timers

- Line mode: LIST command
- Panel mode: TIMER command

3-27

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The NetView LIST TIMER command displays AT, EVER, and AFTER timers. The NetView TIMER command displays timers on a full-screen panel. You can also use TIMER to create or modify timers. In addition, TIMER supports the display of CHRON timers.

LIST TIMER example



LIST TIMER example

```

DISPLAY OF OUTSTANDING TIMER REQUESTS
TYPE: CHRON  TIME: 09/04/07 16:01:55
COMMAND: CHRON AFTER= 00:03:00 RECOVERY=IGNORE ROUTE=AUTOAON
COMMAND= 'CNMENVHB PLEX1,TIVED2,NETA.AOFDB.APPL.NETVIEW'
PGMAT=2007-09-04-16.01.55.402849
OP: AUTOAON (AUTOAON) ID: SYS00006           TIMEFMSG: NO    GMT
TYPE: EVERY   TIME: 09/04/07 16:03:50      INTERVAL: 000 00:02:00 EVERYCON: YES
COMMAND: MEMSTORE 5% 5
OP: AUTO2 (AUTO2) ID: MEMSTORE           TIMEFMSG: NO    GMT
TYPE: CHRON  TIME: 09/04/07 16:10:07
COMMAND: CHRON AFTER= 00:10 GMT ID=FKX00007 RECOVERY=IGNORE ROUTE=
AUTDVIPA COMMAND='FKKEDVPA AUTDVIPA' PGMAT=
2007-09-04-16.10.07.581168
OP: AUTDVIPA (AUTDVIPA) ID: FKKEDVPA        TIMEFMSG: NO    GMT
TYPE: AFTER   TIME: 09/04/07 16:10:32
COMMAND: FKXEACT3 TIVED1 RUTTCP2
OP: AUTTCP2 (AUTTCP2) ID: FKKEDVPA        TIMEFMSG: NO    GMT
TYPE: EVERY   TIME: 09/04/07 16:10:57      INTERVAL: 000 00:10:00 EVERYCON: NO
COMMAND: IDLEOFF 60 521
OP: AUTO1 (AUTO1) ID: IDLEOFF           TIMEFMSG: NO    GMT
TYPE: AFTER   TIME: 09/04/07 16:50:30
COMMAND: FKXEACT2 TIVED1 TIVED1 SP
OP: AUTOAON (AUTOAON) ID: FKKEDVPA        TIMEFMSG: NO    GMT
TYPE: AT      TIME: 09/04/07 23:59:59
COMMAND: MSG ALL,MIDNIGHT - All 2nd shift operations can go home
OP: AUTO1 (NETOP2) ID: MIDNIGHT          TIMEFMSG: NO    LOCAL
7 TIMER ELEMENT(S) FOUND FOR ALL
END OF DISPLAY

```

LIST TIMER=ALL,OP=ALL: Displays all timers for all NetView tasks

3-28

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Use the LIST TIMER command to display active timers. You can display timers for a specific task, all timers, or a specific time.

The slide, which shows seven timers, is an example of a LIST TIMER=ALL,OP=ALL command to display all timers for all NetView tasks. Field characteristics are as follows:

- The *TYPE*: field indicates the ID type of timer: two EVERY, two AFTER, two CHRON, and one AT.
- The *OP*: field indicates the task that the timers are scheduled under: AUTO1, AUTO2, AUTOAON, AUTTCP2, and AUTDVIPA.

To display the timers for AUTO1 task, issue the following command:

```
LIST TIMER=ALL,OP=AUTO1
```

To display only the MEMSTORE timer that is scheduled under AUTO2, issue the following command:

```
LIST TIMER=MEMSTORE,OP=AUTO2
```

The MEMSTORE timer monitors the usage of several NetView files (for example, EXECs and DSIPARM members). MEMSTORE loads the most frequently used files in NetView storage.

If you have many timers scheduled, you can use the TIMER command. The IDLEOFF timer monitors NetView tasks. If a task is inactive, it can be logged off.

TIMER command

TIMER command

- Use of panel interface to manage NetView timers:
 - Display
 - Modify
 - Delete
 - Create
- Managing of timers in remote NetView domains:
Uses RMTCMD over SNA or TCP/IP
- Support of AT, EVERY, AFTER, and CHRON
- Alternative to using LIST TIMER command



3-29

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The TIMER command displays AT, EVERY, AFTER, and CHRON timers on a full-screen panel and can be used for deleting, modifying, or creating timers. The TIMER command can be used for displaying timers on remote NetView systems that use RMTCMD over SNA or TCP/IP. If you run SA z/OS, the TIMER command also uses the SA z/OS XCF communication between NetView domains.

TIMER display example



TIMER display example

```

EZLK6000          TIMER MANAGEMENT      AOFDA TSCCW01 07/08/11 06:27:07
Target: AOFDA    Target Network ID:      Operid: TSCCW01 Selected: 3
IP Addr: _____   _____                   Purged: 0
Port: _____      Remote Target Date and Time: _____
Filter criteria:
Type one action code. Then press enter.
1|A=Add 2|C=Display/Change 3|P=Purge 4=Add CHRON timer
  Timer ID Scheduled Type Interval Task Save Catchup
  - MEMSTORE 07/08/11 06:27:32 EVERY 00:02:00 AUTO2
  - MEMSTORE 5% 5
  - IDLEOFF 07/08/11 06:29:32 EVERY 00:10:00 AUTO1
  - IDLEOFF 60 593
  - XCFTMR$2 07/08/11 06:29:33 CHRON 00:05:00 AUTOXCF
  CNMEXCON TYPE=CHCKCONN

Command ==>
F1=Help F2=End F3=Return F5=Refresh F6=Roll
F7=Backward F8=Forward F11=Reset Target F12=Cancel
3-30

```

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The slide shows an example of the panel resulting from a TIMER command. Operators can work with timers in other NetView domains by specifying the following information:

- Domain ID in the *Target*: field for TCP/IP communication
- Network ID in the *Target Network ID*: field for SNA communication)

The EZLRMTIMER statement in CNMSTYLE controls communication between NetView domains as follows:

- EZLRMTIMER=NETV: Systems use the RMTCMD command between them over TCP/IP or SNA.
- EZLRMTIMER=SA: Systems use SA z/OS XCF communication to SA z/OS NetView domains and RMTCMD over SNA communication to others.



Tip: The *Remote Target Date and Time* field can be helpful if you connect to a NetView domain in a different time zone.

After purging (deleting) a timer, an additional PF key (PF9=Purged Timers) is shown on the panel. Press PF9 to display timers that have been purged. You can recover one or more of the purged timers.

Creating an AT timer



Creating an AT timer

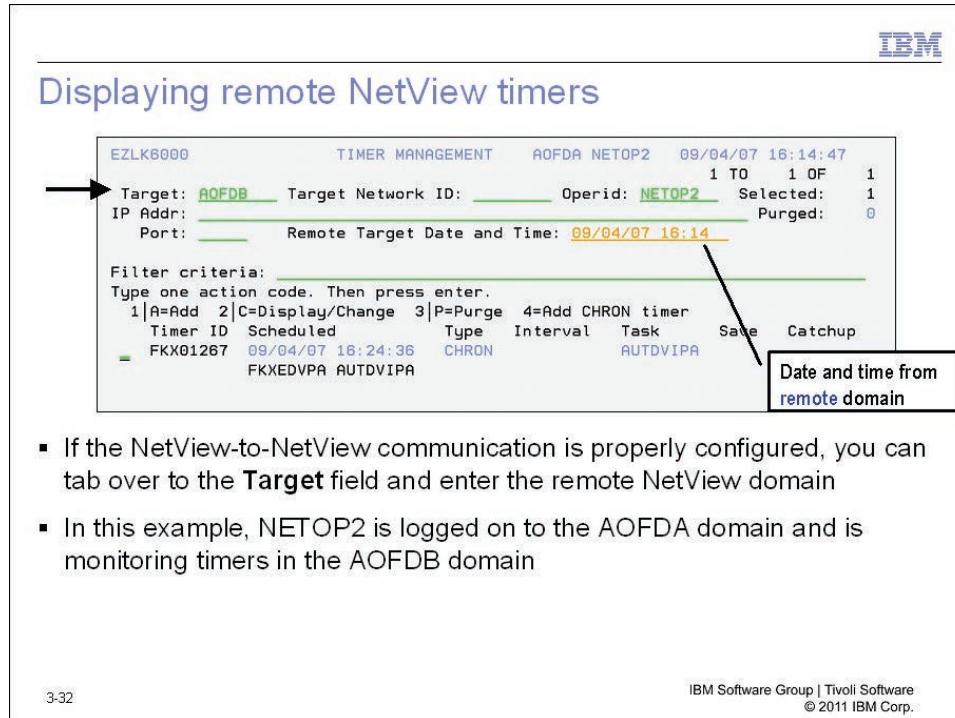
```
EZLK6120      Set AT timer      AOFDA NETOP2 09/04/07 15:55:39
Target: AOFDA  Target Network ID: _____ Operid: NETOP2
IP Addr: _____
Port: _____ Remote Target Date and Time: _____
Timer Type  2 1 EVERY : ..... : AT
             2 AT   : ..... :
             3 AFTER : ..... :
             4 CHRON: Time Format (HH:MM:SS) :
TIMEFMSG .. 1 No 2 Yes : Time . 23 : 59 : 59 :
Timerid . . MIDNIGHT : Date Format (MM/DD/YY) :
Task . . . AUTO1 : Date . 09/04/07 :
Save . . . 1 No 2 Yes : ..... :
Scheduled .
Timer Command MSG ALL,MIDNIGHT - All 2nd shift operations can go home.
```

1. Fill in *Timer Type* (AT), *Timerid* (MIDNIGHT), *Task* (AUTO1), *Time Format* (23:59:59), and *Timer Command* (MSG ALL)
2. Press ENTER to obtain the following text:
EYL973I REQUESTED TIMER MIDNIGHT ADDED ON AOFDB

Command ==> F1=Help F2=End F3=Return F6=Roll F12=Cancel

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Displaying remote NetView timers



Notice the *Remote Target Date and Time* field. In this example, the local and remote NetView domains are in the same time zone.

Displaying the list of remote domains

IBM

Displaying the list of remote domains

This example shows the following information:

- TIMER command is issued in the AOFDA domain
- The AOFDB domain is connected with operator TSCCW01

EZLK5500 REMOTE TARGET SELECTION 1 to 2 of 2
Filter: _____
Type one action code and press enter.
■ DOMAIN SYSTEM SYSPLEX COMM NETID OPERID PORT VERSION
 AOFDA IP Addr: 10.31.187.194 n/a USIBMES 4022
 AOFDB IP Addr: 10.31.187.195 TCP/IP USIBMES TSCCW01 4022 V6R1
Command ---->
F1=Help F2=Forward F3=Return F4=Refresh F5=Roll
F7=Backward F6=Cancel

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

If you are unsure of the NetView domains that you can connect to, type a question mark (?) in the *Target* field of the TIMER panel to see a panel similar to the slide.



Note: The domain list is built from RMTSYN statements and also any active RMTCMD sessions (SNA and TCP/IP). You might have other NetView domains that are capable of receiving a RMTCMD session that are not in the list. They might not be in the list because they are not defined with RMTSYN or are not currently active.

In this example, the TIMER command can connect to the AOFDB domain by using RMTCMD over TCP/IP. This panel also shows an existing connection to the AOFDB domain as TSCCW01.

Filtering timer data

IBM

Filtering timer data

Timer ID	Scheduled	Type	Interval	Task	Save	Catchup
FKX00009	09/04/07 16:20:09	CHRON		AUTDVIPA		
		FKKEDVPA	AUTDVIPA			
FKX00010	09/04/07 16:20:33	AFTER		AUTTCP2		
		FKXEACT3	TIVED1	AUTTCP2		
FKX00003	09/04/07 16:50:30	AFTER		AUTORON		
		FKXEACT2	TIVED1	TIVED1 SP		

3-34

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can use the *Filter criteria* field to reduce the number of timers that are displayed on the TIMER panel. In this case, NETOP2 filters for all timers with a timer ID that begins with FKX.

Purging timers



Purging timers

- Line mode: PURGE command
PURGE TIMER=timer_id,OP=operator_id
- Panel mode: TIMER command
Option three (3 | P=Purge)

3-35

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Use the PURGE TIMER command to delete timers. If you do not know the timer ID or the task, use the TIMER command to display a list of timers and select option three to delete a timer.

Student exercise

IBM

Student exercise

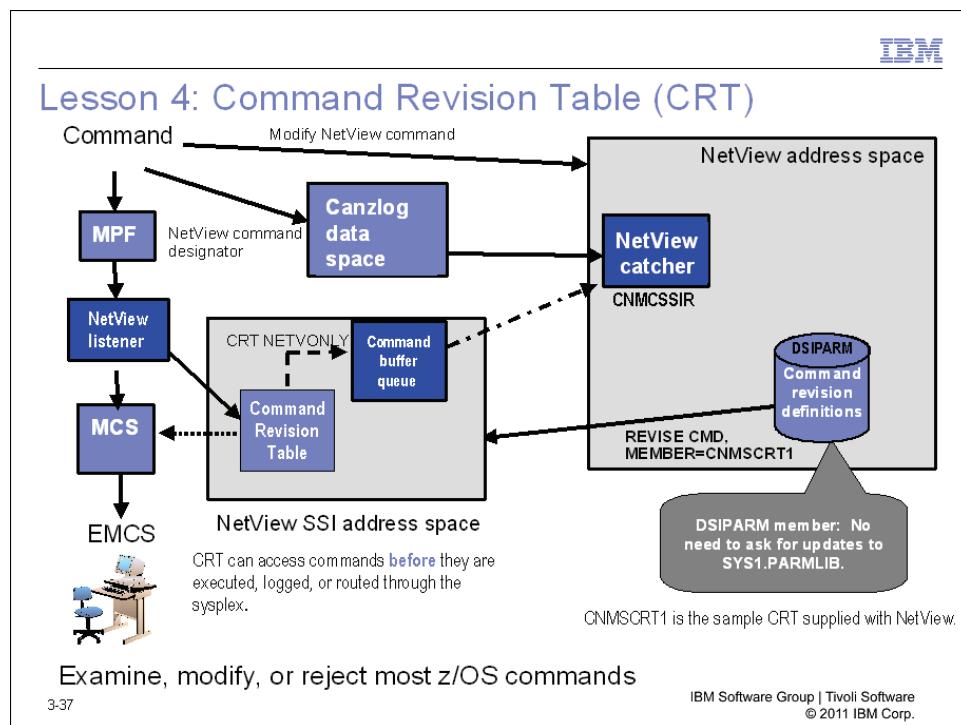


3-36

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 3-3.

Lesson 4: Command Revision Table (CRT)



In NetView for z/OS 5.4, the MVS Command Management was deprecated and replaced by Command Revision Table. With the Command Revision Table (CRT), you can intercept MVS commands before they process. Command sources include the MVS console and the NetView MVS command.

The CRT intercepts any text that is entered on an MVS console command line. The text might be as follows:

- An SDSF system command
- A command using the JCL COMMAND parameter
- From a program using the MGCRE macro or direct SVC 34

The text entered might or might not be a valid MVS command before it is altered or redirected by CRT processing.



Note: If a command is reissued altered or unaltered by means of the REISSUE command as part of CRT processing, that command is exempt from CRT action. Issuing a command by using a NetView MVS command does not pass control to the CRT.

You can change the command text, write a message to the command issuer, and run the command, or suppress the command. You can also transfer the command to the NetView program for more detailed actions. The CRT can remain active even while the NetView program is not, but the SSI address space is required.

The syntax of the Command Revision Table is very similar to the syntax of the MRT. To enable the CRT, you must enable the MPF DSIRVCEX exit. This exit is in the SCNMLNKN library. This library must be in the LINKLIST. See the *IBM Tivoli NetView for z/OS Automation Guide* for more details.



Note: Commands that the CRT's NETVONLY statement routes to NetView make use of the SSI address space. Commands with a NetView command designator are routed through the Canzlog data space. Commands that are sent to NetView through a MODIFY command are retrieved directly from z/OS.

Student exercise

Student exercise



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 3-4.

Summary



Summary

Now that you have completed this unit, you can perform the following tasks:

- Implement the NetView-to-NetView communication required for multisystem automation
- Route commands to NetView operators and automation tasks
- Use NetView timers to proactively monitor resources

3-39

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Unit 4: Managing the NetView automation table



Unit 4:

**Managing the NetView automation
table**



© 2011 IBM Corp.

Introduction

This unit discusses the basics of the NetView automation table and how to manage one. The unit also includes information about multiple automation tables.

Objectives

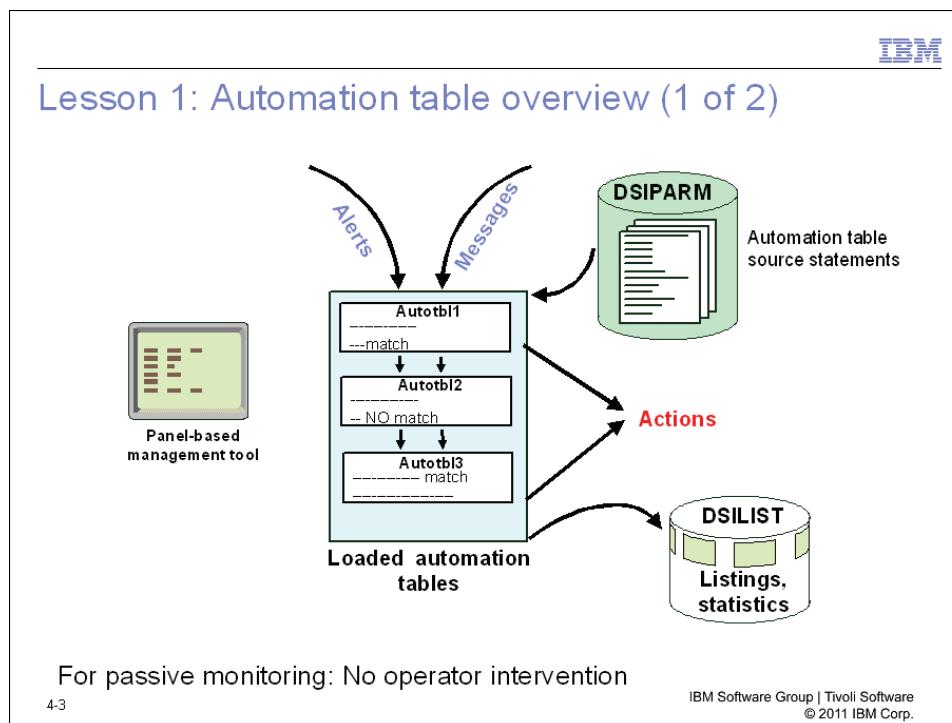


Objectives

When you complete this unit, you can perform the following tasks:

- Describe the basic functions provided by the NetView automation table
- Describe the structure of the automation table
- Manage the automation table
- Describe how to improve the efficiency of an automation table

Lesson 1: Automation table overview



The automation table identifies events (messages and MSUs) to trap, actions to take, and where to schedule those actions. This unit focuses on the management of the automation table. Discussion of the syntax of the statements is in Unit 5: Coding an automation table.

The automation table is a set of IF-THEN statements that you can use for trapping messages and alerts and taking actions. The table might consist of one member or a combination of several members. The table is in your DSIPARM data set.

The automation table is the key to *passive automation*. As messages and SNA Management Services Unit (MSU) events enter NetView, they route to the active automation tables for matching. If a match for a message or MSU is within a table, an action might be taken as a result. Automation tables can be tested before use, and the results written to a data set (DSIARPT).

Statistics pertaining to the use of the table are available by using the AUTOCNT command. Output from this command can be written to a member of the DSILIST data set for further analysis.

A panel-driven command (AUTOMAN) can be used for managing the automation tables that are in use.

Automation table overview (2 of 2)



Automation table overview (2 of 2)

- Contains statements to trap events, take actions, and route the actions to appropriate tasks
 - Primary use is for passive (reactive) automation
 - Take Action is based on occurrence of an event
- You can modify statements only if you use an editor, such as ISPF (no tool available to modify from NetView)
- NetView supplies an automation table: DSITBL01
- NetView also provides management and statistical tools

Automation table basic approach

IBM

Automation table basic approach

**Source statements
in DSIPARM**

```
IF <condition1> THEN <action1> ;  
IF <condition2> THEN <action2> ;  
IF <condition3> THEN <action3> ;
```

- Set of IF-THEN statements
 - Located in members of NetView DSIPARM
 - Each statement ends with a semicolon
- For each event that enters the automation table:
 - Test each IF <condition> clause for a match
 - If a match is found
 - Take Action indicated (THEN <action> clause)
 - o Issue command or REXX procedure
 - o Route to an automation task
 - o Set other parameters to control logging, display, and so on

4-5

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

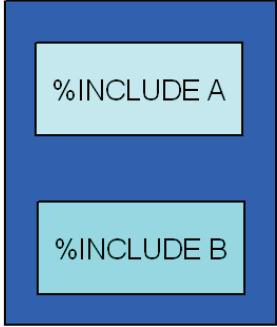
The basic approach is testing each message that arrives in NetView against the IF clause of each statement of the table in sequence. If a match is made, the actions that are indicated in the THEN clause occur. Automation processing is complete for that message. Any IF-THEN statements after the match are ignored for that message. The basic approach has exceptions.

The simplest automation table approach is having a single automation table that has only IF-THEN statements. The single table is a single member of the DSIPARM data set.

Segmented automation tables

Segmented automation tables

Automation table



- One logical table that is loaded and searched:
For example, AUTTBL01
- Sequential search within a table
- Possibility of containing one or more members:
 - %INCLUDE identifies the members
 - Each member can contain
 - Complete IF-THEN statements to check for messages or MSUs
 - SYNonyms
 - Actions
- Possible use: Each group within an organization can maintain their own automation table

4-6

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

An automation table can, like many NetView definitions, be built up from many separate members. It can be coded by using %INCLUDE statements for other DSIPARM members, each containing table statements. Thus, you can create one logical table from many physical elements. A member that is included can contain %INCLUDE statements for other members.

Such a merged table is checked, loaded, and searched as one logical unit. Testing for a message or MSU starts with the first statement in the (logical) table and progresses by testing each statement in sequence until a match occurs. This basic approach has exceptions.

Multiple automation tables

IBM

Multiple automation tables

- Possibility of several automation tables being active
- Search still sequential:
 - Processing controlled based on
 - Occurrence of a match
 - CONTINUE() statement
 - Table order controlled at load time
- CONTINUE() statement:
 - YES: Processing continues with next statement in current member
 - NO: (default) Processing of current member ends and begins with first statement in next member
 - STOP: Ends all processing
- CONTINUE(YES): Capability of impacting automation table performance

```

    graph TD
        MSG((WARNING)) --> TBL1[AUTOTBL1]
        TBL1 -- "Match found" --> TBL2[AUTOTBL2]
        TBL2 -- "Match found" --> TBL3[AUTOTBL3]
        TBL3 -- "Because of STOP, AUTOTBL3 is not searched" --> STOP[STOP]
    
```

4-7

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Multiple logical automation tables can be active at any time. When several tables are active, they are managed in sequence. When a message or MSU arrives, the first table processes it. When the processing in that table completes (either a match, or reaching the table end), processing begins at the start of the next table. This process repeats until all tables are processed. The order is the same for all messages and MSUs.

Exceptions to this sequential order are as follows:

- CONTINUE(STOP) action: Indicates that all automation table processing ends, even if later tables have not been processed.
- CONTINUE(YES) action: Indicates that automation table processing continues with the next statement in the current member. If not used carefully, overuse of CONTINUE(YES) can impact the performance of your automation table.

Multiple tables are useful for reducing the complexity of managing a single table. With a single table, it could be difficult to ensure that a statement processes for the intended message. By using multiple tables, you can initially develop and test smaller units, then implement them with confidence.

Lesson 2: Automation table management



Lesson 2: Automation table management

NetView provides several commands to assist you with managing your automation tables

- AUTOTBL
- AUTOMAN
- AUTOTEST
- AUTOCNT

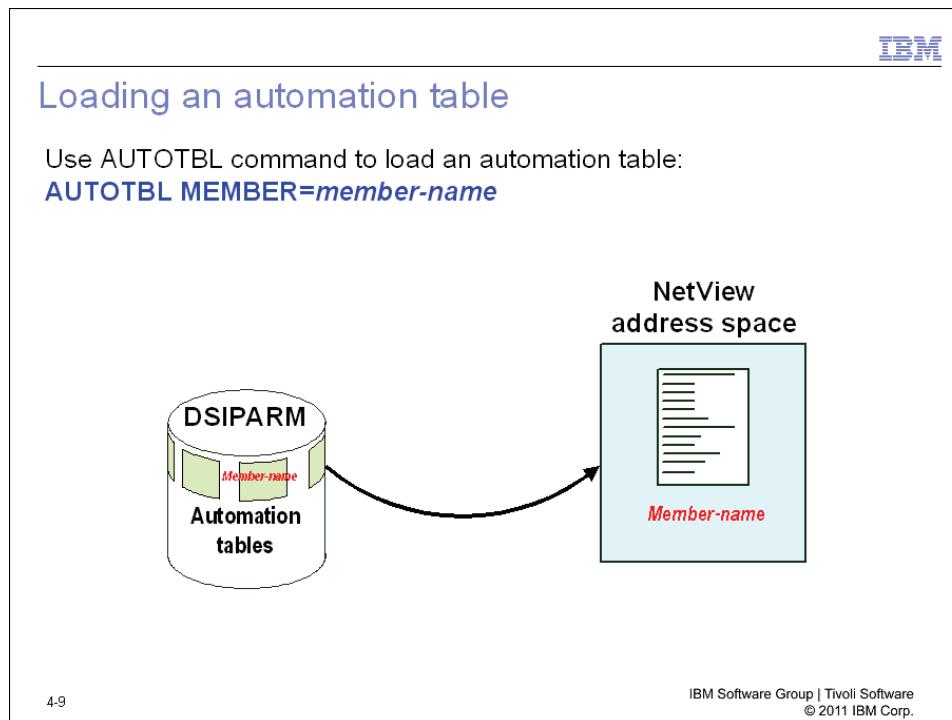
4-8

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Use the AUTOTBL and AUTOMAN commands to load and manage the NetView automation table. Example uses are as follows:

- Use the AUTOTEST command for testing changes to an automation table against messages stored over a typical day.
- Use the AUTOCNT command for displaying statistical information for tuning an automation table.

Loading an automation table



Use the AUTOTBL command for activating, testing, and checking statuses of automation tables. The basic command for activating a single automation table is as follows:

AUTOTBL MEMBER=member-name

Use the AUTOTBL command for the additional following purposes:

- Inserting, removing, and replacing individual automation tables
- Loading additional tables into the list of active tables
- Displaying the statuses of tables
- Disabling and enabling automation table statements or groups of statements
- Listing details of automation tables

NetView can load automation tables during initialization based on CNMSTYLE statements. Discussion about loading automation tables from CNMSTYLE occurs later in this lesson.

Table activation results in resolution of %INCLUDE, >%IF, and SYN statements, then the creation of in-storage table definitions in a form that can yield higher performance.



Note: Automation table members are in DSIPARM and can be loaded in storage with the memStore statement. If you change an automation table and do not see the effects of the change, issue the LIST MEMSTOUT command to see if your table is loaded in storage.

AUTOTBL examples

IBM

AUTOTBL examples

- Examples
 - AUTOTBL STATUS
 - AUTOTBL OFF
 - AUTOTBL MEMBER=MYTBL01
 - AUTOTBL MEMBER=MYTBL01,TEST
 - AUTOTBL MEMBER=MYTBL01,LISTING=MYLIST
 - AUTOTBL REMOVE,NAME=YOURTBL
- Notes
 - TEST does not produce a listing file
 - LISTING produces a listing file in DSILIST

4-10

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can test an automation table before using it. You can find common syntax errors before attempting to load the table into production. If the table has errors, the load fails and all other updates (including those from co-workers) do not load. A table that has invalid statements or BEGIN-END sections also do not load.

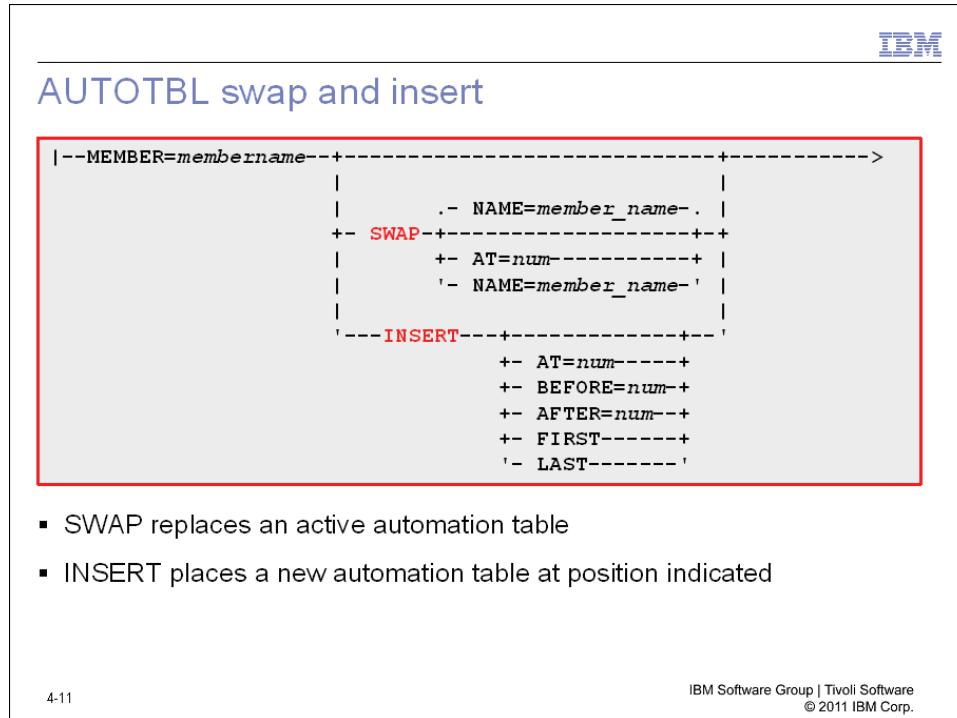
While a table reloads, automation is always available. The new table is building while the old one is still being used. Afterward, the new one is switched into use after it finishes building.

The AUTOTBL listing helps in correcting errors. It provides an expanded copy of the automation table with all synonyms resolved. The listing is written to a member of the DSILIST data set when the LISTING keyword is specified.

Use the name parameter to specify an entire table within a list of active automation tables. You can enable or disable individual statements or a group or block of statements.

You can use a panel interface to issue the AUTOMAN command to load and manage your automation tables.

AUTOTBL swap and insert



If you run more than one automation table, two important operands of the AUTOTBL command are as follows:

- SWAP: Used for replacing an active table with a new member. If the same name is used, this action is effectively a reload in place. If the current active table has any disabled statements, similar statements do not disable when the new table loads.
 - INSERT: Used for placing a new table within the active table as indicated, the number specified indicating its position.

FIRST and **LAST** do not require use of a number and can be removed only by using the **REMOVE** option.

Automation table load notes



Automation table load notes

- Does not load for the following conditions:
 - Invalid statement
 - Invalid BEGIN-END sections
- Active automation table continues running when updating, reloading, or testing
- You perform a test of automation table before loading

4-12

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

If an automation table contains syntax errors, it does not load. If an active automation table already exists, it remains the active table. The current automation table is replaced only when the new table does not contain any errors.

By default, the automation table does not load if you call commands or EXECs that do not exist. For example, suppose you create automation table statements to issue an EXEC called RECOVER but have not created the EXEC when the table loads. This can result in an error unless you set a constant in the NetView DSICMOD constants module. As an alternative, you can create the RECOVER EXEC with only a comment line.

Loading an automation table from CNMSTYLE

IBM

Loading an automation table from CNMSTYLE

- Use CNMSTYLE statement AUTOCMD
Automation tables load when NetView initializes
- Load the NetView-supplied automation table:
 - AUTOCMD.DSITBL01.list = ListName
 - AUTOCMD.DSITBL01.marker = (AON)
 - AUTOCMD.DSITBL01.order = ***FIRST***

Using an order of ***FIRST*** locks DSITBL01 as the first automation table if there are others.
- Load a user automation table (MYTABLE) for generating a listing file of MYLIST:
 - AUTOCMD.mytable.LIST = MYLIST
 - AUTOCMD.mytable.MARKER = XYZ
 - AUTOCMD.mytable.ORDER = C

Remaining ORDER statements control the order in which automation tables are loaded (sorted alphanumerically).

4-13

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The automation table can be loaded as follows:

- Manually with the AUTOTBL or AUTOMAN commands
- When NetView initializes

Use the AUTOCMD definition statements in CNMSTYLE to load automation tables when NetView initializes. The syntax is as follows:

```
AUTOCMD.member-name.LIST = listfile
AUTOCMD.member-name.marker = marker
AUTOCMD.member-name.order = A | B | C ...
```

- *Member-name* is the automation table member name in DSIPARM. An example is DSITBL01 or MYTABLE.
- *Marker* associates the table with descriptive text that can be displayed when the table status is queried. For example, NIGHT can identify the table for third-shift operations.
- *Order* identifies the position that the table is to load in. ***FIRST*** locks the member as the first automation table. The tables load based on an alphanumeric sort of the order statement.

AUTOTBL STATUS example

IBM

AUTOTBL STATUS example

```
* AOFDA      AUTOTBL STATUS
' AOFDA
BNH361I THE AUTOMATION TABLE CONSISTS OF THE FOLLOWING LIST OF MEMBERS:
AOFDAPPT COMPLETED INSERT FOR TABLE #1: DSITBL01 AT 09/04/07 15:49:47 (FIRST)
NETOP2    COMPLETED INSERT FOR TABLE #2: PJQTBL01 AT 09/05/07 09:07:49
AOFDAPPT COMPLETED INSERT FOR TABLE #3: VAPLAT   AT 09/04/07 15:49:47
```

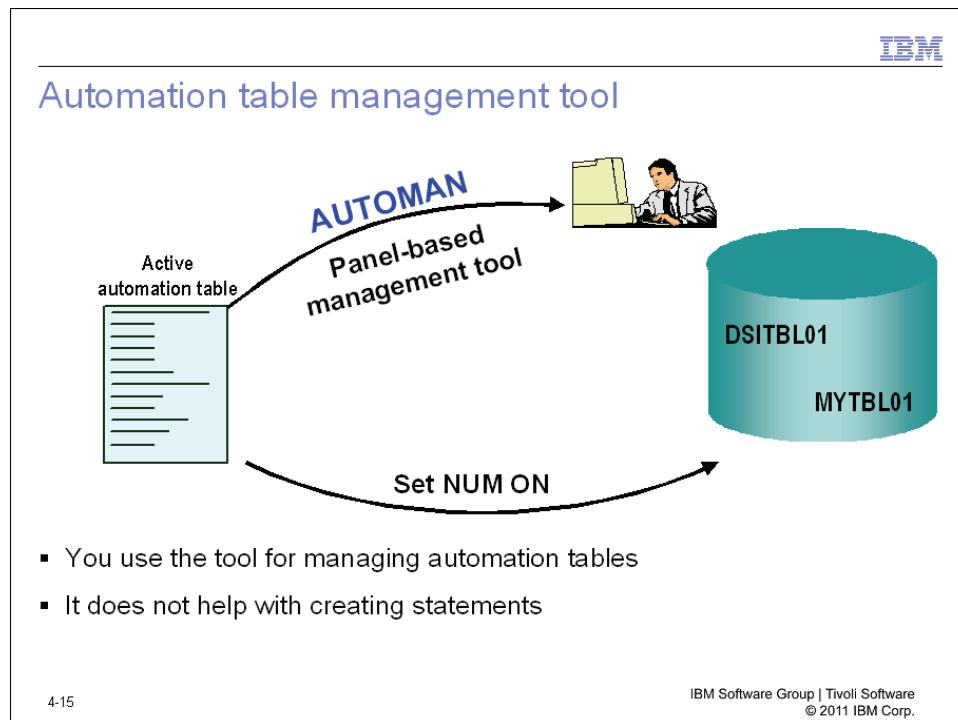
- Three automation tables are active: DSITBL01, PJQTBL01, VAPLAT
- DSITBL01 is the first automation table
- DSITBL01 and VAPLAT are loaded when NetView initializes:
Task AOFDBPPT

4-14

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

AUTOTBL STATUS displays the status of the automation tables. In this example slide, three automation tables are active. DSITBL01 is locked as the first table. PJQTBL01 is in position two and is loaded by NETOP2 after NetView initialized. All tables that loaded during initialization are loaded under the PPT task.

Automation table management tool

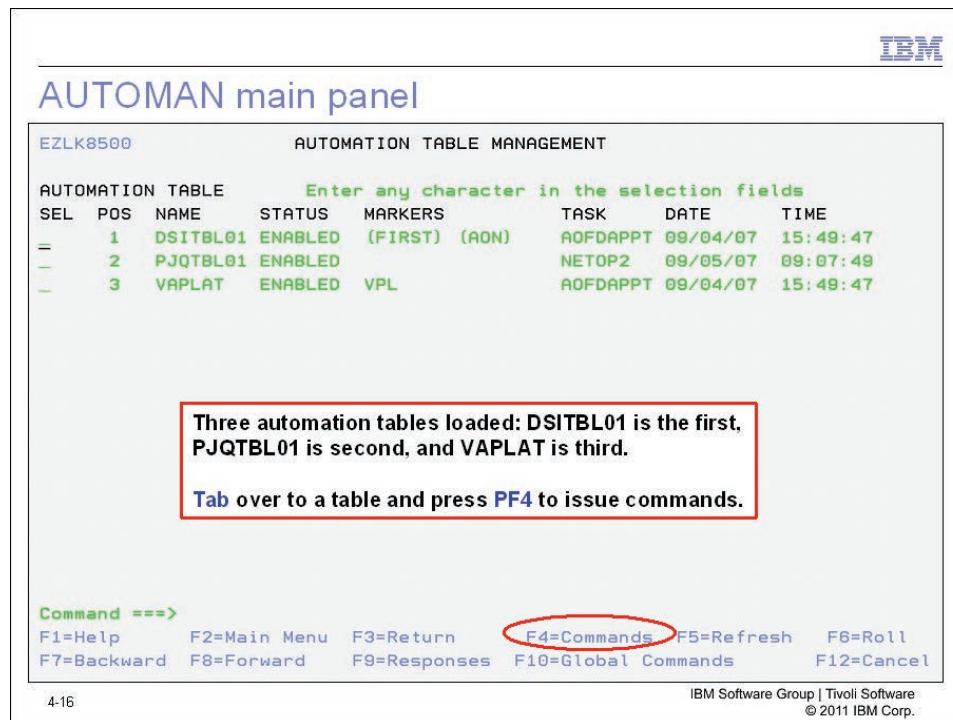


NetView supplies an automation table management tool called AUTOMAN, which is a command that displays a color-coded panel to assist with the following tasks:

- Table loading, reloading, and removing
- Table and member insertion
- Enabling and disabling parts of a table
- Browse table or listing (full or partial)
- Viewing of table statistics

AUTOMAN is an alternative to the AUTOTBL command, but some actions require that the table statements have sequence numbering. (Use TSO EDIT to set NUM ON.)

AUTOMAN main panel



The AUTOMAN command displays the automation table management tool, which permits all AUTOTBL commands to be issued. At invocation, it produces a panel similar to the slide example.

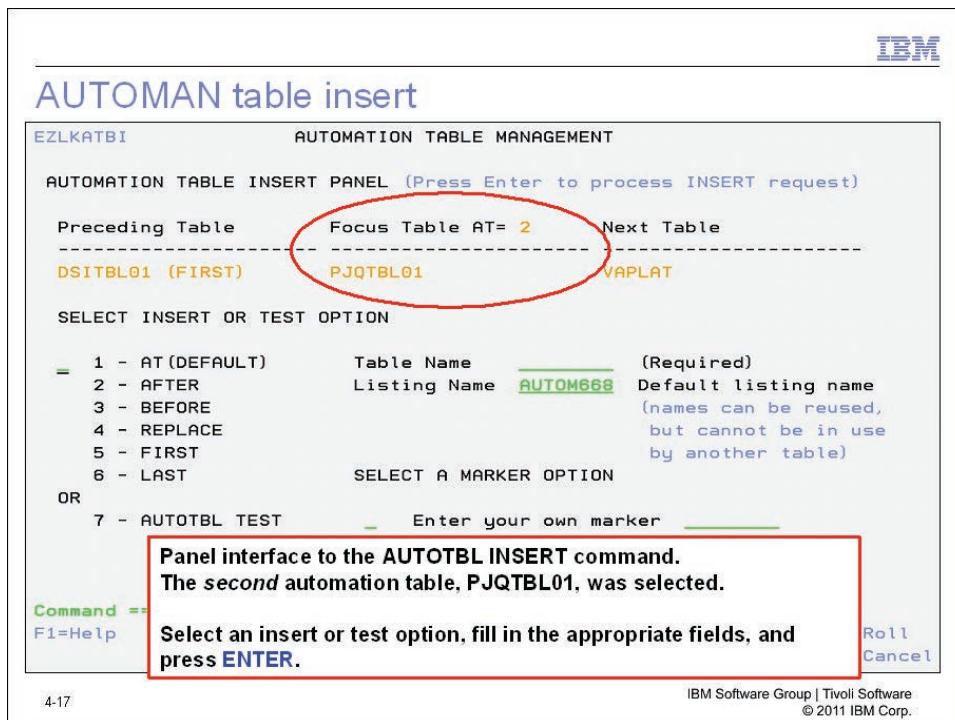
The initial display shows all loaded tables in their current order. By default, the NetView automation table, DSITBL01, is *marked* as the first table. If you have the AON component active, it indicates that AON is active and using DSITBL01.

To take an action against a member, perform the following steps:

1. Tab over to the SEL column for that member:
2. Type any non-blank character, and press ENTER.
3. Press PF4 (Commands).

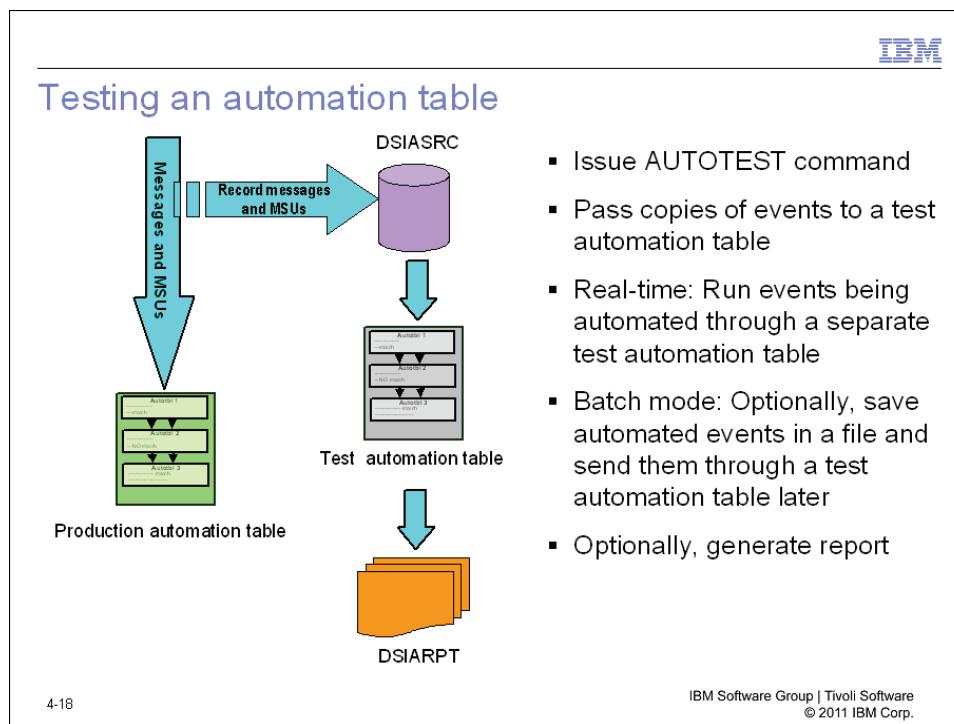
Press PF9 to display a panel that contains the results of any tests or error messages.

AUTOMAN table insert



This example shows how to insert an automation table member by using AUTOMAN. The current member (focus table) is PJQTBL01. Select an insert or test option. For example, option **2** (AFTER) inserts a new table after PJQTBL01. Specify the table name in the *Table Name* field, and press **ENTER**.

Testing an automation table



You can use the AUTOTEST command to perform the following tasks:

- Create a member that contains the messages and MSUs that entered the production automation table.

You can use these messages later as input to the test automation table. The collection does not require the test automation table to be active. It does not affect regular automation table processing.

- Discover and correct any logic, typographical, or ordering problems before putting an automation table into the production environment.
- Compile a test automation table and load it into storage in preparation for activating an automation table test.
- Activate or deactivate the testing of a test automation table.
- Specify the source of messages and MSUs to be used as input to the test automation table being tested.
- Display the status of the test automation table testing function.

Displaying automation table statistics



Displaying automation table statistics

- AUTOCNT display of summary or detailed statistics for an automation table: Messages, MSUs, or both
- Statistics, which can be logged to a file
- Usable to determine if a command has been issued
- Statistics are reset when the automation table is loaded or AUTOCNT RESET is issued
- Examples
 - To display summary statistics for messages and MSUs for the default (first) automation table:
AUTOCNT REPORT=BOTH,STATS=SUMMARY
 - To display detailed statistics for messages for a specific automation table:
AUTOCNT REPORT=MSG,STATS=DETAIL,NAME=DSITBL01

4-19

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

An automation table must be designed correctly and then tuned. To assist with tuning, NetView provides the AUTOCNT command. The AUTOCNT command creates performance statistics pertaining to the active automation tables. These statistics can be written to a member of the DSILIST data set for further analysis.

The statistics give information about the flow of messages through the automation table. Data is included for the average number of matches that are made for each message. Additional information about each statement includes the number of compares are made for it, and how matches there are. With the statistics, a user can determine whether the table is well-constructed and efficient or not.

Statistics are removed when a table reloads or if the AUTOCNT RESET command is issued.

AUTOCNT example

IBM

AUTOCNT example

NetView V6R1M0		Tivoli NetView		AOFDA NETOP1		07/11/11 15:06:56							
* AOFDA		AUTOCNT REPORT=MSG,STATS=DETAIL											
* AOFDA													
DW08001 AUTOMATION TABLE MSG DETAIL REPORT BY NETOP1													
----- (DSITBL01/LISTNAME MESSAGE DETAILS 07/11/11 15:06:56) -----													
AOFDAPPT COMPLETED INSERT FOR TABLE #1: DSITBL01 AT 06/30/11 13:12:59													
-- PERCENTAGES --													
STMT	L	SEQUENCE NUMBER/ MEMBER	COMPARE	MATCH	E C A D	MATCH/ COMP	COMP/ MATCH						
NUMB	I	LABEL NAME	NAME	COUNT	C I I I	C I C O M P	TOTAL TOTAL						
MEMB			DSITBL01										
0029	S	#0000119	DSITBL01	20731	0 1	0.0	100.0 0.0						
MEMB			CNMSEPTL										
0030	S	#0000021	CNMSEPTL	20731	0 1	0.0	100.0 0.0						
0031	S	#0000025	CNMSEPTL	20731	0 1	0.0	100.0 0.0						
0032	S	#0000029	CNMSEPTL	20731	0 1	0.0	100.0 0.0						
0033	S	#0000033	CNMSEPTL	20731	0 1	0.0	100.0 0.0						
0034	S	#0000049	CNMSEPTL	20731	0 1 X	0.0	100.0 0.0						
0035	S	#0000063	CNMSEPTL	20731	0 1 X	0.0	100.0 0.0						
0036	S	#0000090	CNMSEPTL	20731	0 1	0.0	100.0 0.0						
0037	S	#0000099	CNMSEPTL	20731	0 1	0.0	100.0 0.0						
0038	S	#0000107	CNMSEPTL	20731	0 1	0.0	100.0 0.0						
0039	S	#0000120	CNMSEPTL	20731	1 1 X	0.0	100.0 0.0						
0040	S	#0000134	CNMSEPTL	20731	0 1 X	0.0	100.0 0.0						
0041	S	#0000147	CNMSEPTL	20731	7752 0	37.4	100.0 37.4						
0042	S	#0000152	CNMSEPTL	7752	11 1	0.1	37.4 0.1						
0043	S	#0000154	CNMSEPTL	7741	6381 1	82.4	37.3 30.8						
0044	S	#0000156	CNMSEPTL	1360	816 1	60.0	6.6 3.9						
0045	S	#0000158	CNMSEPTL	544	544 1	100.0	2.6 2.6						
0047	S	#0000166	CNMSEPTL	12979	0 1	0.0	62.6 0.0						
222	***												

4-20

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can also trace the processing of a message or MSU through the automation table using the TRACE action. The TRACE action also sets a trace tag as an indicator that the event is to be traced while the automation table processes the event. Detailed trace information is displayed near BNH370I message for each part of each automation table statement that is processed. See the *IBM Tivoli NetView for z/OS Automation Guide* for more details on debugging tools.

AUTOCNT REPORT=MSG,STATS=SUMMARY example

IBM

AUTOCNT REPORT=MSG,STATS=SUMMARY example

```
NetView V6R1M0          Tivoli NetView   AOFDA NETOP1  07/11/11 15:10:47
* AOFDA    AUTOCNT REPORT=MSG,STATS=SUMMARY
' AOFDA
DW0801I AUTOMATION TABLE MSG SUMMARY REPORT BY NETOP1
----- ( DSITBL01/LISTNAME MESSAGE SUMMARY 07/11/11 15:10:47 ) -----
AOFDAPPT COMPLETED INSERT FOR TABLE #1: DSITBL01 AT 06/30/11 13:12:59
STATISTICS STARTED      = 06/30/11 13:12:59
TOTAL MSGS PROCESSED    = 20738
MSGS MATCHED            = 17637
MSGS RESULTING IN COMMANDS = 8048
TOTAL COMMANDS EXECUTED = 8056
TOTAL ROUTES EXECUTED   = 0
AVERAGE COMPARES/MSG    = 35.27
AVERAGE MSGS/MINUTE     = 1
MINUTES ELAPSED          = 15957
-----
```

4-21

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

If statistics for AVERAGE COMPARES/MSG seem very high, you should consider restructuring your automation table.

Student exercise

IBM

Student exercise



4-22

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 4-1.

Summary



Summary

Now that you have completed this unit, you can perform the following tasks:

- Describe the basic functions provided by the NetView automation table
- Describe the structure of the automation table
- Manage the automation table
- Describe how to improve the efficiency of an automation table

Unit 5: Coding an automation table

IBM

Unit 5:

Coding an automation table



© 2011 IBM Corp.

Introduction

This unit discusses the syntax of the NetView automation table statements. Examples illustrate various automation table coding techniques.

Objectives



Objectives

When you complete this unit, you can code the automation table statements as follows for message automation:

- Parse messages
- Issue commands
- Route the message and commands

Lesson 1: Coding automation table statements



Lesson 1: Coding automation table statements

Several basic questions to consider:

- What do you want to automate?
- What actions do you want to take?
- Where do you want those actions occur?
- How do you want to process the event?

5-3

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Considerations when creating automation table statements are as follows:

- Events to automate: Events can be noted with a single message, multiple messages, a multiline message, an alert and a message, or combination.
- Actions to take: You can choose to issue commands directly from the automation table. You can issue a REXX EXEC and use it to issue the commands and validate their responses. The action to take might be routing the event to a focal point NetView in a multisystem environment.
- Areas to run the actions: By default, the actions run under the primary receiver of the message. That task might not be able to fully support the actions or, there might be no primary receiver.
- Ways to process the event: You could suppress the event. You could have it routed to an operator and held on their screen. You might use other means of processing the event.

Characteristics of events

IBM

Characteristics of events

- Messages
 - Message ID
 - Message prefix
 - Resource name (within message text)
 - Type of message
 - Action, information, error, warning, reply
 - Solicited, unsolicited
 - Multiline message
 - Source of message (NetView domain)
 - Already automated or not
- MSUs (alerts)
 - Alert ID
 - Resource name within the hierarchy
 - Message text subvector
 - Message prefix

5-4

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide lists characteristics of messages and MSUs that you can use for automation. The most common automation is message-based, using the message ID or specific text within the message string. The event that drives the automation table can be a Common Base Event (CBE) or TCP/IP trap that translates into a message or MSU.

All solicited and unsolicited messages are processed on the automation table except the ones as follows:

- Those written directly to DSILOG by the following items:
 - MSG LOG command
 - PIPE LOGTO output
- Intradomain copies created by the following items:
 - ASSIGN SEC or COPY command
 - MSGROUTE command
 - Automation table ROUTE actions

Automation action choices



Automation action choices

- Take no action
- Issue a command
- Issue a NetView PIPE
- Issue a program, such as a REXX EXEC
 - Most common usage
 - Issuing commands, such as the following examples:
 - Collect more data pertaining to the event or resource involved
 - Take corrective actions

5-5

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Most automation actions call a program, such as a REXX EXEC, to perform the following tasks:

- Further interrogate the event. Perhaps it is a multiline message and you have a REXX EXEC to parse the multiline message for a resource name or resource status.
- Issue one or more commands for gathering more information about the event. For example, the event might be a symptom of an previously known problem.
- Issue one or more commands for taking corrective actions, waiting for the response to the command before proceeding to the next one.

Routing automation actions

IBM

Routing automation actions

- Tasks that actions can route to the following types of tasks:
 - Task that receives the event
 - Another task
 - An automation task, for example
 - A task that uses the task name (AUTO1) or nickname (?Primary)
 - Group of tasks
- Considerations for routing actions:
 - Some tasks are not capable of issuing commands:
Examples are NetView Data Services Tasks (DSTs), one task being BNJDSERV
 - Routing actions to an automation task increases likelihood that the action occurs

5-6

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The automation table supports routing the event to one or more NetView tasks by coding the tasks as part of the automation table statement. The routing can go to the following destinations:

- The task that receives the event
- The primary receiver
- A different NetView task (for example, an automation task)
- A group of tasks
- An automation task by means of a *nickname*

The nickname is defined in CNMSTYLE. The task nickname function is similar to a variable. An example is ?Primary.



Tip: Use the QOS command for querying the autotask that is associated with a nickname. For example, to display the autotask known as ?Primary, issue the following command:

```
QOS OPID=?PRIMARY
DWO837I AUTO1 IS DEFINED AND LOGGED ON TO NETVIEW
```

Actions from the automation table are, by default, routed to the task that receives the event. For messages, this task can be the primary receiver. If no primary receiver is defined, the default task

could be the task that started CNMCSSIR for z/OS messages. Or the default task could be the PPT task for VTAM messages.

In the case of alerts and MSUs, the default task for actions is the BNJDSERV task. Because BNJDSERV does not support commands, always code a ROUTE statement for alert and MSU automation.

Processing the event



Processing the event

In addition to executing commands, the automation table supports several actions:

- Display
- Log
- Beep
- Hold
- Change color of message
- And so on

5-7

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Processing the event refers to the handling of the event. Should it be displayed or suppressed? Should it be logged? Should it be held on an operator screen until action is taken to resolve the problem?

Most of these flags are binary, for example, DISPLAY(Y) versus DISPLAY(N).

Automation table statements

Automation table statements

- **IF-THEN:** Include sets of conditions, followed by set of actions if conditions are met
- **BEGIN-END:** Group relevant statements together
Can also be used for improving automation table performance
- **ALWAYS:** Specify actions occur for all events that reach the current point in the automation table
- **%INCLUDE:** Include individual tables or sections of tables
Improves ease of coding and maintenance
- **SYN:** Define synonyms for use later in automation table
- Conditional statements: Include segments that are based on enabled towers or variables



5-8

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can code the following statement types in an automation table:

- IF-THEN statement: A IF-THEN statement tests for specified conditions (characteristics of a message or MSU). If the condition matches the statement, one or more actions can be taken. The IF-THEN statement can also specify where to route the actions.

The IF-THEN statement is the most prevalent statement in an automation table.

- BEGIN-END section: Groups statements for processing.
- ALWAYS statement: Specifies action that occurs for all messages and MSUs that reach this statement.
- %INCLUDE statement: Includes separately coded and maintained sections of the table (so that the table can be built from sections). The included members must also be in DSIPARM.
- Conditional statement: Identified by %>, this statement includes specific statements conditionally. Examples include statements that are used with TOWER definitions that are coded in CNMSTYLE.

Conditional statements are also usable with user-specified entries for creating separate flows through an automation table when shared among multiple NetView domains.

- SYN statement: Defines synonyms for later use in the table.

Automation table statements can be entered in upper or lower case. When mixed-case statements are used, they are converted internally to upper case as this unit shows. Text string comparisons are always case-sensitive. Error messages might display the statement in upper-case format.

Some elements must always be entered in correct case and are not converted. Examples include the following text types:

- Text strings, including message IDs
- Member names that are used in %INCLUDE statements
- Synonym names, which are case-sensitive

IF-THEN overview

IF-THEN overview

- IF <condition> THEN <action> <routing information>;

By default: one match, done,
both of which you can control
- IF-THEN statement identifies as follows:
 - One or more conditions of an event (many condition possibilities)
 - One or more actions to take if the conditions are met
 - Running any valid NetView command
 - Running a program (for example, REXX)
 - Where to route the event and actions
 - Not all NetView tasks are capable of supporting actions
 - You route actions to an automation task
- IF-THEN statement can include nested IF-THEN statements
Must use BEGIN-END structure

5-9

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The IF-THEN statement specifies the following items:

- One or more tests that are to be made against a message or MSU (by using characteristics, such as message ID)
- The actions that are to be taken if the test is true
- The place or places to route the actions
- The way to process the event

The IF-THEN statement can include nested IF-THEN statements when you use the BEGIN-END structure in the automation table.

IF-THEN condition

IF-THEN condition

- Specification of event to intercept and parse
 - Messages
 - MSUs
- Comparison, based on =, ≠, >, <, ≥, ≤
- Conditions combinable by using ampersand (&), or (|), or parentheses
- Encounter of match
 - Actions occur as defined
 - By default, search ends within the current automation table
 - You can code CONTINUE()
 - You might need to code BEGIN-END structure



5-10

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

By default, after a match is found, the search of the current table stops and the search begins with the first statement in any subsequent (logical) table. You can control this process with the CONTINUE() function. If CONTINUE(YES) is coded as part of the action, the search continues at the next statement. Note the following syntax rules:

- Valid comparisons are =, ≠, >, <, ≥, and ≤.
- Tests are combinable by using a logical AND (&) or a logical OR (|) operator.
- Conditions are specified by using a bit string or parse template
- For multiline messages, the parse template can trap only the first non-blank line of the message.(An example later in this unit shows the use of the ACQUIRE function to access the remaining lines.
- Each statement ends with a semicolon (;

Examples of IF-THEN condition items

Function	Description	Messages	MSUs
JOBNAME	MVS originating job	✓	
JOBNUM	MVS assigned job number	✓	
MSGID	Message identifier	✓	
TEXT	Text of message	✓	
TOKEN	Blank delimited token of a message string	✓	
HIER	Resource hierarchy associated with an MSU		✓
DOMAINID	Originating NetView domain	✓	✓
CURSYS	Current MVS system name	✓	✓
HDRMTYPE	Message type	✓	✓
IFRAUSDR or OPID	Originating NetView task name	✓	✓
NVCLOSE	Bit to indicate NetView CLOSE in progress	✓	✓
ROUTECODE	MVS routing codes	✓	
SYSPLEX	Local sysplex name	✓	✓

This tabular list shows a subset of the condition items.
See the Automation Guide for the complete list

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide illustrates a subset of the available condition items for an IF-THEN statement and whether each applies to messages or events. The lesson provides examples of using JOBNAME, MSGID, TEXT, TOKEN, HDRMTYPE, and more. Descriptions of all condition items is in the *Automation Guide* manual.

Examples of IF-THEN actions

Action	Description	Messages	MSUs
DISPLAY	Display the message	✓	
NETLOG	Log message to DSLOG	✓	
SYSLOG	Log message to system log	✓	
BEEP	Sound audible alarm	✓	✓
COLOR	Set foreground color	✓	✓
CONTINUE	Do not end processing at this statement	✓	✓
CNM493I	Specify if audit message should be written to DSLOG	✓	✓
EXEC	Issue command and, with the ROUTE parameter, route message to a task	✓	✓
SRF	Set recording-filter attributes		✓
XLO	Specify external logging		✓
EDIT	Specify an edit specification that alters the AIFR	✓	✓

This tabular list shows a subset of the action items.
See the *Automation Guide* for the complete list

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide illustrates some of the actions that are possible when the automation table receives a message or MSU. Some actions might revert to defaults. Even if a matched automation table statement does not specify an action, an action might occur because of settings that the DEFAULTS or OVERRIDE commands establish. Exact descriptions of actions are in the *Automation Guide* manual.

IF-THEN EXEC action

IBM

IF-THEN EXEC action

```
IF <condition item> THEN  
  EXEC(CMD(' cmdstring ')  
        ROUTE(routeparms));
```

The EXEC action can contain the following items:

- CMD: Issues the specified command
 - Can be any valid NetView command
 - Can include command parameters
Cmdstring = command p1 ... pn
 - Must be enclosed in quotation marks
 - Can have multiple CMD parameters
- ROUTE: Controls the task where the message or command routes to

Note the usage of parentheses

The purposes of the EXEC actions are as follows:

- Issuing commands based on a match to the *condition item*.

The command string must be enclosed in quotation marks. The message text can be parsed and stored into variables to be passed to the command string as parameters.

- Controlling where the commands run.



Note: The EXEC statement must contain an even number of parentheses, commonly overlooked by new users.

IF-THEN ROUTE parameters

IBM

IF-THEN ROUTE parameters

```
-----  
V   I  
<<-ROUTE ( +-----+ * -----+ ) ----->>  
    +- ONE --+    +- PPT -----+  
    '- ALL --'    +- taskname --+  
                  +- +group -----+  
                  '- ?auto -----'
```

- Route the message or cmdstring by using a parameter as follows:
- ONE: Priority is as follows:
 1. The first logged-on operator in the list
 2. The first operator who is assigned to a group in the list and is logged on
- ALL: All operators and groups of operators in the list who are logged on
- *: (Default) The task that the message was received on
- PPT: The NetView PPT task
- Taskname: The specified task
- +group: One or more operator groups
- ?auto: The automation function name of an automation task

5-14

IBM Software Group | Tivoli Software

© 2011 IBM Corp.

This slide explains the possible ROUTE parameters. This type of parameter routes the message and the command to the tasks specified. The most common usage is routing the message or action to one task from a list that can include the following destinations:

- The task that is to receive the message (*)
 - A list of task names (for example, AUTO1 AUTO2)
 - An operator group (for example, +CICSOPS)
 - an *automation function* name

One of the keywords, ONE or ALL, must be specified.



Tip: Use the QOS command to query the autotask that is associated with a nickname, For example, to display the autotask known as ?Primary by using the following code:

QOS OPID=?PRIMARY
DWO837I AUTO1 IS DEFINED AND LOGGED ON TO NETVIEW

IF-THEN ROUTE examples



IF-THEN ROUTE examples

- ROUTE(ONE AUTO1 AUTO2):
 1. Route the message or command to AUTO1
 2. If AUTO1 is not active, then route to AUTO2
- ROUTE(ONE *):Route the message or command to the task that the message was received on
- ROUTE (ONE ?Primary):Route the message or command to the task that is defined with the automation function name ?Primary. By default, the message or command routes to AUTO1 because of
CNMSTYLE: function.autotask.primary = AUTO1

5-15

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

- ROUTE(ONE AUTO1 AUTO2): The message and action route to the first logged-on task between AUTO1 and AUTO2.
- ROUTE(ONE * AUTO1 AUTO2): The message and action route to one of three tasks: the first logged-on operator of: the task receiving the message, AUTO1, or AUTO2.
- ROUTE(ONE *): The message and action route to the task that is to receive the message.
- ROUTE (ONE ?Primary): The message and action route to the task known as the primary task, which is defined in CNMSTYLE as AUTO1.
- ROUTE(ONE +MVSOPS): The message and action route to the first logged-on operator in the +MVSOPS operator group list.

BEGIN-END block

IBM

BEGIN-END block

```
IF <condition> THEN  
  BEGIN;  
    IF <condition1> THEN ...  
    IF <condition2> | <condition3> THEN ...  
    ALWAYS ...;  
  END;
```

- Block of automation table statements:
Similar to most programming DO-END constructs
- Starts with BEGIN action on IF-THEN or ALWAYS statement
- Logically segments automation table
- Can be used for improving performance:
Create functional blocks or subsets of automation table

5-16

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can use a BEGIN-END block to logically segment an automation table similar to using a DO END programming construct.

Use BEGIN-END blocks to improve the performance of an automation table. For example, you can check a message prefix and define a BEGIN-END group to process all messages that match the prefix. If the prefix does not match, not all of the more specific checks (located in the BEGIN-END group) occur:

```
IF MSGID = 'EZL' . & TEXT=MESSAGE THEN
BEGIN;
* Process all NetView EZL messages here
...
END;

IF MSGID = 'IST' . & TEXT=MESSAGE THEN
BEGIN;
* Process all VTAM (IST) messages here
...
END;

IF MSGID = 'DSI' . & TEXT=MESSAGE THEN
BEGIN;
* Process all NetView DSI messages here
...
END;
```

ALWAYS statement

ALWAYS statement

- Specify actions or series of statements to be performed for all events that reach the specification point in the automation table
- For setting defaults, use ALWAYS with CONTINUE at start of the automation table
- Use ALWAYS with BEGIN-END structure:
 - Use for setting defaults for a BEGIN-END section of the automation table
 - Use at the end to process events that do not match in the BEGIN-END:
Similar to REXX OTHERWISE statement
- Use ALWAYS at the end of automation table to process events that do not match at all:
For example, keep an inventory of the messages that were not automated, and catch a message for you to automate



5-17

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

For all messages and MSUs that reach the specification point in the automation table, specify either actions or a series of statements that always process. You can use an ALWAYS statement with the CONTINUE operand at the start of the automation table or a BEGIN-END section for setting defaults for that table or section. Doing so sets an action for all messages and MSUs that reach that point.

Code an ALWAYS statement at the end of a table for processing all messages and MSUs that have not been trapped in that table. Examples are as follows:

- Defining an automation table defaults for logging and continuation:

```
ALWAYS SYSLOG(Y) NETLOG(Y) CONTINUE(Y);
```

- Defining an automation table default for CNM493I message:

```
ALWAYS CNM493I(Y);
```

IF-THEN example 1

IBM

IF-THEN example 1

```
IF MSGID = 'DSI039I'  
    THEN DISPLAY(Y) HOLD(Y) BEEP(Y) NETLOG(N)  
        SYSLOG(Y);
```

DSI039I MSG FROM NETOP1 : HELLO

If message DSI039I is detected, use the automation table as follows:

1. Display the message
2. Hold the message (with high intensity)
The message does not scroll off NetView screen
3. Sound the audible alarm for the terminal
4. Do not write the message to the NetView log (DSILOG)
5. Write the message to the system log

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Message DSI039I indicates that a message has been received from a NetView task using the MSG command. The following actions are to occur:

1. Display the message to the recipient at their NetView screen.
2. Hold it so that it does not flow off the screen as further messages arrive.
3. Sound the terminal alarm as the message is displayed.
4. Write the message to the system log but not the network log, DSILOG.
5. Do not issue a command.

IF-THEN example 2

IBM

IF-THEN example 2

```
IF TOKEN(1) = 'DSI374A' & DOMAINID = 'CNM01'  
    THEN EXEC(CMD('CLISTA')) ROUTE(ONE AUTO1 AUTO2));
```

DSI374A THRESHOLD REACHED, *threshnum* BUFFERS ON MESSAGE QUEUE OF *taskname*

- If the first token of the message is DSI374A and the message comes from NetView domain CNM01, CLISTA runs:
 - CLISTA runs under one operator
 - Message routes to the first active operator in the list: AUTO1, AUTO2
- If both tasks are inactive, the following actions apply:
 - CLISTA does not run
 - Message DWO032E goes to DSILog

5-19

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

In this example, the IF clause contains a test for the first token of the message. In most cases, this is the same as the message ID. An exception is a WTOR message, where the first token is the reply number, and the message ID is the second token.

Two tests are made against the message (TOKEN(1)='DSI374A' and DOMAINID='CNM01'). Both conditions need to be true in this case. You can make many tests and combinations by using AND or OR conditions.

In this example, the action is running a command named CLISTA. This could be a REXX EXEC, command list (Clist), or any valid NetView command.

The ROUTE operand designates the task where the command runs. The ROUTE operand specifies the ONE keyword, which indicates that the command runs on only one of the tasks listed. The tasks test in the order listed. If an operator is active, the command is scheduled to run on that operator task.

In this example, if the AUTO1 task is active, the command runs only there. If AUTO1 is not active, CLISTA runs under the AUTO2 task if it is active. If neither task is active, the command does not run. The DWO032E message is written to the NetView log.

If ALL is specified instead of ONE, CLISTA runs on each active task in the list. An alternative approach is to specify ROUTE(ONE * AUTO1 AUTO2). The action then routes to the task that receives the message first, known as the *primary receiver* for the message.

IF-THEN example 3

IBM

IF-THEN example 3

```
IF MSGID = 'IEF404I' &
  (JOBNAME='J123' | JOBNAME='J124')
  THEN EXEC(CMD('PRODJOB'));
```

IEF404I job_nam - ENDED - TIME=hh.mm.ss

- When job J123 or J124 ends (message IEF404I), the PRODJOB command runs
- If no ROUTE command is specified, PRODJOB runs under the primary receiver of the message

5-20

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This example includes more complexity in the set of statements. The test is for message ID IEF404I, which occurs for only jobs J123 or J124.

The action causes the PRODJOB command to run. No operator is specified with a ROUTE, which causes the PRODJOB command run under the task that receives the IEF404I message, known as the primary receiver, which is almost always designated to be an active autotask.

You can achieve same effect if the CMD clause has a ROUTE(ONE *) specification. In this case, the * indicates the assigned primary operator.

IF-THEN example 4

IBM

IF-THEN example 4

```
IF TEXT = . 'CLOSE' . & TEXT(9) = 'DSILOG'.
    THEN EXEC(CMD('CLISTB') ROUTE(ALL +GRP1));
```

- If the message text contains the character string CLOSE anywhere in the message text
(Note the use of the two periods as placeholders.)
- and
- If the message text also contains the string DSILOG in position 9 to 14

Run CLISTB for ALL tasks that are currently assigned to the +GRP1 NetView operator group

5-21

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

In this example, the actual text of the message is checked instead of a message ID. The message text is checked for two character strings, CLOSE and DSILOG, using the TEXT conditional:

- **TEXT = . 'CLOSE' . :** Using the periods before and after the character string specifies that the word CLOSE can be *anywhere in the message string*. In other words, any text can precede or follow the string CLOSE.
- The following examples contrast the period positions:
 - . 'CLOSE' specifies that the message must *end* with the character string.
 - 'CLOSE' . specifies that the message must *begin* with the character string.
- **TEXT(9) = 'DSILOG' . :** Tests the text of the message starting at character position nine. This is checked for the string DSILOG. The period character that follows the string DSILOG is called a placeholder. The placeholder indicates that the rest of the text (to the end of the message) can be any set of characters and is ignored.

If the period is omitted, the test tests the message, starting at position 9 is DSILOG, with nothing following.

In this example, the THEN clause requests the CLISTB command run on ALL of the operators in operator group +GRP1. Routing actions to operator groups can be useful because +GRP1 might be a group whose members change with time. For example, the group might contain first-shift operations. If none of the members in the group are active, or if the group is empty, no command runs. The DWO032E message is issued.



Important: With this example, a STOP TASK=DSILOG command produces two messages that drive the automation table and matches this IF-THEN statement. This statement also compares against every message that contains the character string CLOSE.

```
STOP TASK=DSILOG
DSI660I STOP TASK ISSUED FOR DSILOG BY NETOP1
DSI661I STOP TASK ISSUED FOR DSILOG BY NETOP1
DSI531I 'DSILOG' : 'DST' IS TERMINATING
DWO520I DSILOG : VSAM DATASET 'CLOSE' COMPLETED, DDNAME =
'DSILOGP' RETURN CODE = X'00', ACB ERROR FIELD = X'00'
DSI556I DSILOG : VSAM DATASET 'CLOSE' COMPLETED, DDNAME =
'DSILOGP' RETURN CODE = X'00', ACB ERROR FIELD = X'00'
DSI200I TASK DSILOG HAS TERMINATED
```

The DWO520I and DSI556I messages result in a match to the IF-THEN statement in this example. To eliminate this result, code an additional test for one of the message IDs. This example shows how to test a message ID, along with the text of a message, to prevent erroneous messages from matching your IF-THEN statement.

To avoid this condition, specify a MSGID to trap as in this following example:

```
IF MSGID = 'DSI556I' & TEXT = . 'CLOSE' . & TEXT(9) = 'DSILOG'.
THEN
    EXEC(CMD('CLISTB') ROUTE(ALL +GRP1));
```

IF-THEN example 5

IBM

IF-THEN example 5

```
IF MSGID = 'IFB040I' & TOKEN(6) = 'FULL'  
      & TOKEN(3 6) = 'LOGREC'  
THEN EXEC(CMD('SWITCHIN'))  
ROUTE(ONE AUTO1);
```

IFB040I - SYS1.LOGREC AREA IS FULL

1

2

3

4

5

6

If the first token of the message is IFB040I, the sixth token contains the text FULL, and the third token contains LOGREC beginning at its sixth character

Run the SWITCHIN command under task AUTO1

Note: Tokens are delimited by blanks

In this example, the message text to be tested is as follows:

IEFB040I - SYS1.LOGREC AREA IS FULL

If the message ID is IFB040I and the sixth token is FULL, run the SWITCHIN command under the AUTO1 task. The TOKEN operand is used for searching for a blank-delimited field in a message.

To check for the exact composition of part of a token, you can use TOKEN(*x y*). For example, you can add an additional test of TOKEN(3 6) = 'LOGREC' to the preceding IF clause. This tests for the third token to contain exactly the characters LOGREC, starting in position six of the token.

The hyphen (-) after the message ID counts as the second, blank-delimited token. Also, the entire SYS1.LOGREC string is the third token because it contains no blanks.

IF-THEN example 6

IBM

IF-THEN example 6

```
If msgid = 'IEE362A' &
    text=.FOR SYS1.MAN'lib 'ON' volser then
exec(cmd('SMFDUMP 'lib','volser)
        route(one ?smfoper));
```

IEE362A SMF ENTER DUMP FOR SYS1.MANX ON CPDLIB

- Search all IEE362A messages:
 - All text before FOR SYS1.MAN is to be ignored
 - The text between SYS1.MAN and ON (value of X) is assigned to a variable called LIB. Any text after ON (value of CPDLIB) is assigned to a variable called VOLSER
- Run SMFDUMP command, passing it the two variables that are parsed from the message: LIB and VOLSER
- Run the command (SMFDUMP X,CPDLIB) under the ?smfoper task: Defined in CNMSTYLE with function.autotask.smfoper = AUTO1

5-23

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This example shows how variable information is extractable from the message text, used for forming part of the command that is requested in the EXEC action. This example uses mixed-case text.

The IF clause selects a message (IEE362A) that indicates a SMF data set is full. The name of the data set is in the message text.

The TEXT keyword is used for parsing the message text and extracting information from it. Strings that are enclosed in quote marks must be matched. Text between them is assigned to either a placeholder (indicated by a period) or to a variable (identified by a variable name). The name should not have quote marks around it.

In the example, two variables, *lib* and *volser*, are parsed from the IEE632A message and passed as operands to the SMFDUMP command. The blanks that separate these operands must be explicitly specified, along with the command and its parameters. That is, all commands are constructed by appending strings and variables.

The command string ('SMFDUMP 'lib','volser) is built as follows:

1. Beginning with the command name (SMFDUMP), including a blank at the end to act as a delimiter for the first parameter
 2. Appending the *lib* variable as the first parameter
 3. Appending a comma as a delimiter (,) for the SMFDUMP command

4. Appending the *volser* variable as the second parameter

Quotes are required in the command string only when using text strings. In this example, the command name (SMFDUMP) includes the blank and the comma that separate the command parameters.



Tip: The command routes to the ?smfoper. CNMSTYLE defines that ?smfoper is defined in CNMSTYLE as the AUTO1 task. You do not need to hard code the task name if you use the *function* name. If you use a function name, you do not have to change the automation table if you change the task name.

IF-THEN example 7

IBM

IF-THEN example 7

```
Variable assignment
↓
IF MSGID = 'ICH802D' & TOKEN(1) = repnum
    THEN EXEC(CMD('MVS REPLY 'repnum',Y')
              ROUTE(ONE *)) ;
```

nn ICH802D REPLY 'Y' OR 'N'

- Message ICH802D is a RACF message that requires a reply of Y or N
- If message ICH802D is detected, place the first token (the reply number, nn) in a variable called repnum
- Run the MVS REPLY command, using the reply number that is stored in the repnum variable with a response of Y
- Issue the reply on the task that received the message

5-24

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This example shows how to parse a WTOR message and assign the reply number to a variable. In this case, the ICH802D message, is similar to the following code:

33 ICH802D REPLY 'Y' OR 'N'

Because the reply number precedes the message ID, the reply number is in token one of the message string and the message ID in token two. The reply number is stored in a variable called REPNUM. A variable has no quotes (').

ROUTE(ONE *) indicates the MVS REPLY command is to be run by the task that the ICH802D message is assigned to. The REPNUM reply ID passes as an operand to the command. An example MVS REPLY command is MVS REPLY 33,Y.

All blanks in the command that are to run must be explicitly coded. There are no implicit blanks as there are in REXX statements. In this case, the command string is as follows

```
Token 1: MVS
Delimiter: blank
Token 2: REPLY
Delimiter: blank
Token 3: repnum variable
Delimiter: comma
Token 4: Y
```

IF-THEN example 8

IBM

IF-THEN example 8

```
IF MSGID = 'DSI008I' & NVCLOSE ¬='1'  
    THEN EXEC(CMD('RESTART') ROUTE(ONE AUTO1)) ;
```

DSI008I 'OPER99' NOT ACTIVE

If message DSI008I (task not active) is received and NetView is not in shutdown mode, issue the RESTART command under AUTO1

5-25

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The NetView DSI008I message indicates a task is not active. NVCLOSE is a single bit flag that indicates if NetView is shutting down. Bit strings might also be compared. In this case, a test is for a condition that does not pertain to the message itself but to the current environment.

If the message ID is DSI008I and the flag NVCLOSE is not set (NetView is not shutting down), schedule the RESTART command. This command is to run on the AUTO1 task.

IF-THEN example 9

IBM

IF-THEN example 9

```
IF MSGID='IST530I' & TOKEN(9) = 'SY4CDRM'
  THEN NETLOG(Y 3 +MYGROUP)
      SYSLOG(YES)
  CONTINUE(YES);
```

IST530I *runame PENDING FROM fromnetid TO tonetid FOR fornodename*

If IST530I message is received for SY4CDRM resource, the following applies:

- Log the message to DSLOG, setting the STATMON message indicator three for all operators in the +MYGROUP group
- CONTINUE(YES) allows the message to continue to be scanned by subsequent automation table statements
- Code CONTINUE(YES) only when you know that the automation table has additional entries for the message

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This example shows how to use CONTINUE(YES) to continue searching the automation table. If an IST530I message is for a VTAM resource called SY4CDRM (contents of token 9), log the message and set Status Monitor indicator three. The message goes to all operators that are assigned to the +MYGROUP operator group that are currently logged on. You must log the message to set an indicator. *IST530I* is the first line of a multiline message. Discussion about accessing data in the remaining lines of a multiline message occurs later in this lesson.

By default, the automation table processing ends with this match. CONTINUE(YES) allows processing to continue after the match. Any EXEC requests are scheduled, and processing continues with the next automation table statement.

If a later match in the table contradicts an action, the latest action prevails. For example, if a subsequent match specifies SYSLOG(N), that match is the one that is selected.

IF-THEN example 10

IBM

IF-THEN example 10

```
IF MSGID = 'DSI530I' & TEXT(10) = taskname ''': rest THEN
    EXEC(CMD('CLIST1') ROUTE(ONE AUTO1))
    EXEC(CMD('RUNNING 'taskname) ROUTE(ONE AUTO1))
    EXEC(CMD('CLISTB') ROUTE(ONE AUTO2))
DISPLAY(Y);
```

DSI530I 'DSILOG': 'DST' IS READY AND WAITING FOR WORK

- For message DSI530I, starting at position 10, parse the task name, DSILOG, into a variable called taskname
(Note the end of the task name token and how it is parsed)
- Run multiple commands, processed in order:
 1. CLIST1 on AUTO1
 2. RUNNING on AUTO1 with the taskname variable passed as a parameter
 3. CLISTB on AUTO2

5-27

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This example illustrates how to parse a message that contains single quotes in the message text. In this case, the task name is parsed, beginning at position ten using TEXT(10). Because the length of the task name is variable (up to eight characters long), parse and remove the trailing quote, colon, and the rest of the message as the following text shows:

TEXT(10) = taskname ''': rest

The first and fourth (last) quotes indicate the beginning and end of the text string. The second and third quotes indicate the presence of one single quote within the string. The *taskname* insert can be followed by one or more blanks if it has fewer than eight characters. The last character to parse is the colon (:).

This example also runs several commands on AUTO1 and AUTO2 tasks. Commands that are scheduled to the same task run in the specified order. For example, CLIST1 runs and completes on AUTO1 task before the RUNNING task.

CLISTB is scheduled to run on the AUTO2 task. It runs independently of the commands that are scheduled on AUTO1. CLISTB might run before CLIST1 starts, while CLIST1 runs, or even after CLIST1 finishes.

The automation table EXEC action schedules, or queues, a command to run on a task. Only one command can run at a time on a task at a time. The running of the command is said to be *asynchronous* from the processing of the message.

IF-THEN example 11

IBM

IF-THEN example 11

```
IF MSGID = 'BNH367E' & OPID = Toper  
  & TEXT(66 3) = '704' THEN  
    EXEC(CMD('MSG 'Toper 'must use SWAP or INSERT  
              parameter'))  
    DISPLAY(N);
```

BNH367E UNABLE TO COMPLETE AUTOTBL MEMBER REQUEST. REASON CODE: 704

The BNH367E message contains a return code that indicates why an AUTOTBL command failed

- Parse the return code, starting in position 66 for a length of three characters
- Save the operator ID in a variable named Toper
- If the return code is 704, issue a message to Toper by using the MSG command on the task that received the BNH367E, suppressing the BNH367E

5-28

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This example shows two parts as follows:

- Assigning the contents of an automation table function to a variable
 - Reducing confusion that arises when using placeholders or periods within strings

In this example, the OPID function is used for storing the operator ID to a variable, TOPER. TOPER is in the command string. The BNH367E message contains a return code that identifies whether or not an operator is permitted to issue an AUTOBL request. Several ways of accessing the reason code token are as follows:

- Use TEXT(66 3)='704' to test the contents of the message string to confirm if the reason code is 704. The starting position is at 66 for a length of three characters, You can also specify TEXT(66)= '704' and omit the character length.
 - Use the TOKEN() function: TOKEN(10)='704'
 - Parse the text within the message: TEXT = . 'REASON CODE: 704'

The MSG command has mixed-case text as follows:

DSI039I MSG FROM NETOP1 : must use SWAP or INSERT parameter

Automation of multiline messages

IBM

Automation of multiline messages

IEE115I	13.40.16	2007.037	ACTIVITY	767				
JOB	M/S	TS	USERS	SYSAS	INITS	ACTIVE/MAX	VTAM	OAS
00003	00014	00000		00028	00008	00000	/00300	00009
TSO NOT FOUND								

- Automation table can process multiline messages that have ACQUIRE condition:

```
IF MSGID='IEE115I' &
    ACQUIRE('FINDLINE 4 WORD 2.2')='NOT FOUND' &
    ACQUIRE('FINDLINE 4 WORD 1')='TSO' THEN
```

- Most common implementation is checking for the title line message (IEE115I) and calling a REXX EXEC to process the complete message. An example is calling a REXX EXEC to take action

5-29

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This example shows automation that is based on the contents of a multiline message, using the ACQUIRE condition item. The ACQUIRE condition item uses a PIPE EDIT stage to examine data being automated. ACQUIRE is usable for examining any line in a multiline message.

In this example, an MVS D A,TSO command has been issued. Line four of the IEE115I multiline message contains the text, TSO NOT FOUND. If TSO is not active, an action can be taken (for example, an MVS START command) to activate it. This example uses two ACQUIRE condition items to check the status of TSO. An alternative is checking WORD 1.3 for TSO NOT FOUND, using only one ACQUIRE condition item.

Using ACQUIRE avoids the need to create a REXX EXEC for processing the message and issuing a command. All of the logic is contained in the automation table and referenced.



Note: If you need to issue commands for gathering more information, you still need to code a REXX EXEC.

For more information about the PIPE EDIT stage, see the *IBM Tivoli NetView for z/OS Programming: Pipes* manual.

BEGIN-END example

IBM

BEGIN-END example

```
IF SYSID = 'SYSA' THEN
  BEGIN;
    ALWAYS DISPLAY(YES) COLOR(GRE) CONTINUE(Y);
    IF MSGID = 'IEF4' . & JOBNAME = jnam THEN
      BEGIN;
        IF MSGID= 'IEF450I' THEN
          EXEC(CMD('JABEND ' jnam));
        IF MSGID = 'IEF403I' THEN
          EXEC(CMD('JSTART ' jnam));
        IF MSGID = 'IEF404I' THEN
          EXEC(CMD('JENDED ' jnam));
        ALWAYS ;
      END;
    END ;
  
```

Coded this way, the BEGIN-END is similar to a REXX SELECT-WHEN-OTHERWISE statement.

Each IF is similar to a WHEN.

The ALWAYS acts as the OTHERWISE.

- Improves the performance of the automation table by using BEGIN-END:
- If the message is not from SYSA, all statements shown here are skipped
- If the message is from SYSA but not an IEF4 message, the IF MSGID statements are skipped

5-30

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This example shows how to specify BEGIN-END blocks, which can be nested. Use BEGIN-END blocks to reduce processing. For example, a conditional set of statements for SYSA is included. These statements are loaded and processed in all automation tables on all systems, but are effective when run on only the SYSA system. On systems other than SYSA, only the SYSID check (first IF-THEN clause) is performed. The remaining statements in the BEGIN-END block are ignored, improving performance of the automation table.

Automation table variables (*jnam* in the example) are usable by all actions within the BEGIN-END block.

Testing for unsolicited messages

IBM

Testing for unsolicited messages

```
IF MSGID='IST619I'  
  & TEXT = . 'ID = ' resname ' FAILED' .  
  & HDRMTYPE = 'Q'  
THEN  
  EXEC(CMD('EZLEFAIL OPTION=SA TBLKEY=IST619I '  
    ' SKIP=(R) RESNAME='RESNAME )  
    ROUTE(ALL *));
```

- **HDRMTYPE='Q'** identifies a **VTAM** message as *unsolicited*
- Messages that are issued for operator commands are solicited
- Automation is performed on unsolicited messages to recover a failing resource

5-31

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

In some cases, you might want to automate only unsolicited messages. These messages identify actual events that have occurred, such as a VTAM resource failure, a database error, or an application abend.



Note: Most customers do not automate solicited messages. Solicited messages are the direct results of an operator (or automation operator) command.

All unsolicited VTAM messages have a HDRMTYPE message type of Q. You can test for that message type in the automation table, similar to the example. (HDRMTYPE documentation is in the *Automation Guide* and *Programming: Assembler* books.)

You can use the automation table functions to test AIFR bits for determining if the message is unsolicited or not. Use IFRAUIND and IFRAUIN2, bit 16 to indicate whether the message was unsolicited (1) or solicited (0).

Testing for commands



Testing for commands

```
IF HDRMTYPE = 'C' THEN  
    CONTINUE(STOP);
```

- NetView commands drive the automation table:
From operators, REXX execs, automated actions, and so on
- Causes processing to occur:
 - The command text to be checked against the IF-THEN statements that you coded in the automation table
 - Possibility of entire automation table scanned with no match
 - Waste of CPU cycles
- Check for NetView commands (**HDRMTYPE = 'C'**) at the beginning of your automation table. Exit by using a **CONTINUE(STOP)** to prevent automation of the commands

5-32

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Commands that NetView operators issue also drive the automation table. You can improve the overall performance of your automation table by coding a test for the commands at the beginning of your automation table. HDRMTYPE=C indicates that the event is a command from NetView or a message from a REXX EXEC running in NetView (using the SAY instruction).

Use code and action of CONTINUE(STOP) to prevent automation of the commands.

SYN statement

IBM

SYN statement

```

SYN %PRODSYS% = 'SYSID = ''SYSA'' | SYSID='SYSB';
SYN %SHOWOPS% = 'EXEC(ROUTE(ALL MAINOP1 MAINOP2)) HOLD(Y)';
.
.
IF MSGID = 'IEF450I' & %PRODSYS% THEN %SHOWOPS% ;

```

**R
e
p
l
a
c
e
s**

- Define synonyms for use in automation table:
 - Define near top of automation table for ease of use
 - Put synonyms in a separate member and %INCLUDE
 - Reference later in automation table statements
- Use descriptive names to make statements readable
- Use shorthand for long, repetitive strings
- Maintenance: Only one change necessary

```

IF MSGID = 'IEF450I' & SYSID='SYSA' | SYSID='SYSB'
    THEN EXEC(ROUTE(ALL MAINOP1 MAINOP2)) HOLD(Y);

```

5-33 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The SYN statement defines synonyms for use in the automation table. With synonyms, you can use shorthand for long, repetitive strings, which can help with maintenance and modification of an automation table. With synonyms, you can have a single specification of values that appear frequently in subsequent automation table segments. These values might vary in different systems or with changes in environment.

The syntax is as follows:

```
SYN %synname% = 'synvalue';
```

- *Synname* can have as many as 256 characters.
- *Synvalue* is the value of the synonym.
 - The value must not contain a semicolon.
 - You must exercise caution if a value contains quotes.

Additional significant information about synonyms are as follows:

- Must have a name and a value.
- Must be defined before it is used.
- Cannot be redefined later in the table (for example, in a later automation table segment), even with the same value.

When an automation table is loaded, the synonym value is substituted within that table wherever the name is encountered. Synonyms do not carry over to subsequent logical tables.

Conditionally including statements

Conditionally including statements

%>IF statements are usable for selectively including items as follows:

- Statements:

```
%>IF TOWER('SA') THEN  
    IF MSGID='IEF404I' THEN EXEC (CMD('NEWJOB'));
```

- Blocks of statements:

```
%>IF TOWER('SA') THEN BEGIN;  
    IF MSGID='IEF404I' THEN EXEC (CMD('NEWJOB')) ;  
%>END;
```

- Segments:

```
%>IF CURSYS() = 'SYSA' THEN  
%INCLUDE SYSAMSGS
```

%> uses
Data-REXX
support in NetView

- Variable inclusion:

```
'%INCLUDE ' CURSYS() || 'MSG$'
```

5-34

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can conditionally include statements by using the %>IF statements. These statements designate actual inclusions in the automation table definition. Statements that are not selected do not load when building the automation table.

Using %> to conditionally include statements makes use of NetView *Data-REXX* functionality. The automation table is one of many NetView members that support Data-REXX. Use Data-REXX to install and customize NetView. For example, by enabling a NetView tower or subtower, you automatically enable all command definitions, automation table statements, operator definitions, and so on.

Statements that are excluded are not displayed when using AUTOCNT reports. Also, they cannot be disabled or enabled.

This slide shows several examples of conditionally included statements. More information follow:

- If the SA z/OS tower is enabled, include a test for message IEF404I.
- If the SA z/OS tower is enabled, include a BEGIN-END section.
- If the current system is SYSA, include the SYSAMSGS member. You could also use the CURSYS() function as a variable when including SYSAMSGS.

All uses of these REXX functions require that the member read from DSIPARM and have a first line as follows that identifies the member as a *Data-REXX* program:

```
/* %DATA REXX
```



Note: The use of Data-REXX is not restricted to automation tables. All NetView DSIPARM members except CNMSTYLE can use this feature.

CNM493I message

IBM

CNM493I message

CNM493I DSITBL75 : 00440013 : MVS REPLY 13,C

- NetView logs a CNM493I message to identify that a command has been issued, based on a match to an automation table statement:
 - Provides an audit trail for automation actions (in DSILog)
 - In this example, member DSITBL75 at sequence number 00440013 issued an MVS REPLY command
- Default is generating CNM493I messages:
 - You can change the default in CNMSTYLE or with DEFAULTS or OVERRIDE commands to save CPU cycles
 - CNMSTYLE: DEFAULTS.CNM493I = No
 - Command: DEFAULTS CNM493I=NO
 - You can use line numbers if no sequence numbers have been set

5-35

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

A CNM493I message is written to the NetView log when a command is scheduled as a result of automation table processing. In the NetView log, the CNM493I message precedes the message that was matched. The CNM493I message includes information, such as the command that was scheduled, and the automation table segment and statement number where the match occurred. The statement number can be a sequence number or line number within the member. CNM493I messages are useful when checking that intended (or unintended) automation matches occurred.

In this example, an MVS REPLY command was issued from the DSITBL75 member. The command is at the line or sequence number 00440013.

If customers do not want the CNM493I message logged, they can use CNMSTYLE or the DEFAULTS command to disable logging of the CNM493I message. Use CNM493I when you initially develop and test your automation table. Turn off CNM493I when you put your table into production.

You can disable the logging of CNM493I to reduce the amount of messages written to the NetView log. Be careful about disabling. Doing so reduces the information that is available for problem determination, and it can cause confusion.

The format of the CNM493I message is as follows:

CNM493I *member-name* : *sequence number* : *command text*

The command text is shown with variable substitution performed.



Note: If NUM ON has not been issued for the member, the relative line number within the member is shown. In previous releases of NetView, the sequence number would be displayed as N/A if NUM ON had not been set.

Controlling CNM493I



Controlling CNM493I

- You can control logging of the CNM493I message by using the CNM493I() automation table action:
 - CNM493I(N): Do not log a CNM493I message
 - CNM493I(Y): Log the CNM493I message
- ALWAYS CNM493I(N)
Disables logging the message despite the default that you code in CNMSTYLE or specify with a DEFAULTS or OVERRIDE command

5-36

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The CNM493I message action indicates if one of the following CNM493I messages is to be written to the NetView log.

- If a CNM493I action is not specified in the automation table (or by a DEFAULTS or OVERRIDE command), NetView generates CNM493I messages.
- If a CNM493I action is specified, but the appropriate statement has no EXEC action with a CMD keyword, the CNM493I action is ignored.

Student exercise

Student exercise



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 5-1.

Lesson 2: Management Services Unit (MSU)

IBM

Lesson 2: Management Services Unit (MSU)

```
graph TD; MV[Major vector] --- L1[LLLL]; MV --- K1[KKKK]; MV --- SV[Subvectors]; MV --- F1[Final section]; SV --- LL1[LL]; SV --- KK1[KK]; SV --- SF[Subfields]; SV --- F2[Final section]; SF --- LL2[LL]; SF --- KK2[KK]; SF --- D[Data]; SF --- F3[Final section]
```

- Generic term for an SNA Management Services (MS) data record
- Available to NetView to manage and automate
- Structure containing items as follows:
 - Major vectors
 - Subvectors
 - Subfields

 See *SNA Formats* manual for complete details on all vectors

5-38 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This lesson is optional.

The NetView automation table can process both messages and SNA Management Services Units (MSUs). With SNA structure, management information can exchange between Control Points (for example, CP-to-CP, SSCP-to-PU). Such data flows by using generic Management Services Units (MSUs).

A Management Services Unit (MSU) is an SNA data record that is used for providing management information to appropriate points of the network. One point is the NetView program. With such information, the SNA network can be managed at the SNA level (activating control units). NetView uses the data for managing more complexity and automating as necessary.

An MSU is a structured data unit that contains one or more major vectors, which, in turn, contain one or more subvectors. Subfields within a subvector contain the actual data. The size and content of an MSU depends on the Control Point that formats the data.

MSU types

IBM

MSU types

- Five types:
 - Network management vector transport (NMVT)
x'41038D': Flows on SSCP-PU Sessions
 - Control point management services unit (CP-MSU)
x'1212': Flows on LU6.2 MS and HP Transport, PPI
 - Multiple domain support message unit (MDS-MU)
x'1310': Flows on LU6.2 MS and HP Transport
 - Record Maintenance Statistics (RECMS)
 - Record Formatted Maintenance Statistics (RECFMS)
- Note the following items:
 - CP-MSUs and NMVTs contain MSU major vectors
 - MDS-MUs contain CP-MSUs

5-39

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Five types of MSUs, which are envelopes for transporting the actual data, can be automated by NetView as follows:

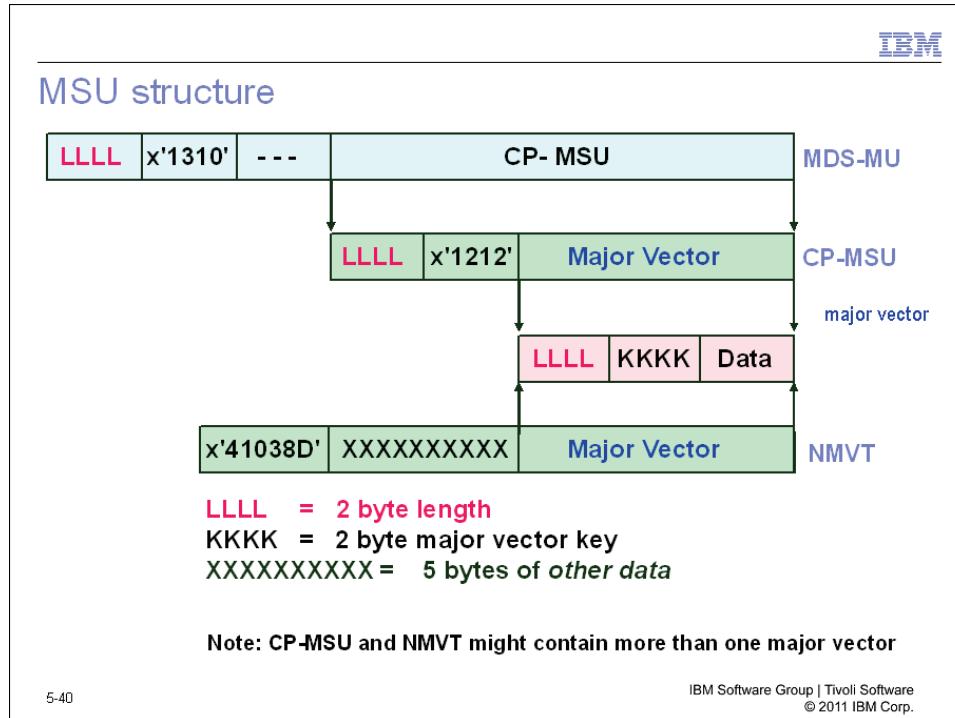
- Network management vector transport (NMVT): Flow on SSSCP-PU sessions
- Control point MSU (CP-MSU): Flow on LU 6.2 MS and HP transport and the PPI
- Multiple-domain support message unit (MDS-MU): Flow on LU 6.2 MS and HP transport between domains
- RECMS
- RECFMS

For automation testing, you can generate alerts with GENALERT.

For online information, enter the following text:

```
HELP GENALERT GENERIC FORMAT
HELP GENALERT NONGENERIC FORMAT
HELP GENALERT RECFMS FORMAT
HELP GENALERT RESOLVED FORMAT
```

MSU structure



An MDS-MU is an envelope that carries CP-MSUs. The CP-MSUs and NMVTs are envelopes that transport data such as major vectors. Detailed descriptions of MSUs are in the *SNA Formats* and the *SNA Management Services Reference* manuals.

Major vectors

IBM

Major vectors

LLLL	KKKK	Subvector	Subvector		
------	------	-----------	-----------	---	---

- Subvectors contain actual data
- Several key major vectors are as follows
 - Alert major vector key = x'0000'
 - Link event major vector key = x'0001'
 - Resolution major vector key = x'0002'
 - PD statistics major vector key = x'0025'
 - Link configuration data major vector key = x'1332'
- Hardware monitor sends major vectors to automation table
- Automation table also processes other MSUs
 - RECMS in a CP-MSU within subvector key x'1044'
 - RECFMS in a CP-MSU within subvector key x'1045'

5-41

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Certain types of MSUs that Hardware Monitor (NPDA) forwards to the automation table are as follows:

- x'0000' Alerts
- x'0001' Link events
- x'0002' Resolution vectors
- x'0025' - PD statistics
- x'1332' Link configuration data

The automation table has access to any MDS-MU or CP-MSU. The original SNA MSU record types are also available to the automation table if they are encapsulated in one of the following subvectors within a CP-MSU:

- Subvector with key X'1044' for RECMS records
- Subvector with key X'1045' for RECFMS records

NetView automation of alert data



NetView automation of alert data

- NetView provides several functions for assisting with automation of alerts:
 - MSUSEG(): Access MSU data
 - MSUSEG(0000.92 8) = Alert ID
 - MSUSEG(0000.31.30 3) = Alert text
 - HIER(): Access alert resource hierarchy data
 - HEX: Compare hexadecimal data strings
- You can use comparable automation table and REXX functions

5-42

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

With the NetView automation table and REXX, you can index into an MSU to test data in its subvectors and subfields.

MSUSEG example 1



MSUSEG example 1

```
IF HMONMSU = '1' & MSUSEG(0000) = ''  
THEN COLOR(GRE) XHILITE(REV),  
SRF (AREC PASS);
```

- HMONMSU=1 indicates an MSU passes from Hardware Monitor (NPDA) to the automation table
- If the MSU contains an alert Major Vector (key of X'0000'), the following actions occur:
 - The alert passes to NPDA for recording (SRF AREC PASS).
 - The color on the NPDA alerts panels changes to green in reverse video

This example shows a test for a Management Services Unit (MSU).

MSUSEG example 2



MSUSEG example 2

```
IF MSUSEG(0000.05.10 5) = HEX('C3D5D4F0F1') .  
  & THRESHOLD (9 0 00:03:00) = '1'  
THEN SRF (ESREC BLOCK) ;
```

- If the MSU contains alert (major vector key of x'0000') and the hexadecimal character string C3D5D4F0F1 in subvector 05, subfield 10, starting in position 5

and

- If it occurs nine times within a three minute period

Stop recording it in the Hardware Monitor (NPDA) data base (SRF(ESREC BLOCK))

Note: In this example, the hexadecimal string C3D5D4F0F1 could be replaced by the CNM01 character string

5-44

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This example shows another test for a Management Services Unit (MSU). Use the period as a placeholder after the hex string to set the limit of the hex string comparison. Days are numeric (zero to 365) and are represented by the value after the threshold count (9 in this case).

SNMP trap automation task



SNMP trap automation task

- An SNMP trap automation task automates an SNMP trap by converting the trap to a CP-MSU and passing that CP-MSU to NetView automation
- Set up the task in CNMSTYLE as a NetView Data Services Task (DST) for receiving and automating SNMP traps
- An SNMP trap automation task can be used for receiving and automating SNMPv1, SNMPv2c, and SNMPv3 traps in both IPv4 and IPv6 networks
- To distribute the SNMP trap automation workload, multiple SNMP trap automation tasks can be used
- Each SNMP trap automation task within an instance of NetView must have a unique task name. However, the same DST initialization member (sample CNMTRAPI) is used for all of them

5-45

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

For more information, see the *IBM Tivoli NetView for z/OS Automation Guide*, Chapter 22, “Automating Messages and Management Services Units,” and Chapter 37, “SNMP Trap Automation.”

Summary



Summary

Now that you have completed this unit, you can code the automation table statements as follows for message automation:

- Parse messages
- Issue commands
- Route the message and commands

IBM Tivoli certification and training

In today's global business world, enhancing and maintaining skills is essential to keeping pace with rapidly changing technologies. Businesses need to maximize technology potential and employees need to keep up to date with the latest information. Training and professional certification are two powerful solutions.

Certification

There are many reasons for certification:

- You demonstrate value to your customer through increased overall performance with shorter time cycles to deliver applications.
- Technical certifications assist technical professionals to obtain more visibility to potential customers.
- You differentiate your skills and knowledge from other professionals and stand out as the committed technical professional in today's competitive global world.

Online certification paths are available to guide you through the process for achieving certification in many IBM Tivoli areas. See ibm.com/tivoli/education for more information.

Special offer for having taken this course

Now through 31 December 2011: For having completed this course, you are entitled to a 15% discount on your next examination at any Thomson Prometric testing center worldwide. Use this special promotion code when registering online or by telephone to receive the discount: **15CSWR**. (This offer might be withdrawn. Check with the testing center as described later in this section.)

Role-based certification

All IBM certifications are based on job roles. They focus on a job a person must do with a product, not just the product's features and functions. Tivoli Professional Certification uses the following job roles used:

- IBM Certified Advanced Deployment Professional
- IBM Certified Deployment Professional
- IBM Certified Administrator
- IBM Certified Solution Advisor
- IBM Certified Specialist
- IBM Certified Operator

Training

A broad spectrum of courses, delivery options, and tools helps keep your employees up to date with the latest IBM Tivoli information:

- *Instructor-led training (ILT)*
Live interaction with an IBM instructor, hands-on lab exercises, and networking with your peers from other companies
ibm.com/tivoli/education
- *Instructor-led online (ILO)*
All the benefits of ILT, but savings on travel dollars and training costs
ibm.com/training/ilo
- *Self-paced virtual classes (SPVC)*
Interactive and hands-on exercises on your schedule
ibm.com/training/us/spvc
- *Web-based training (WBT)*
Training anywhere, any time, that saves you money and travel
ibm.com/training/us/tivoli/wbt
- *Multimedia library*
Modules supporting new and experienced learners with fully animated multimedia clips, step-by-step audio, and companion text
ibm.com/software/tivoli/education/multimedialibrary
- *IBM Education Assistant*
More specific, granular web-based training with individual presentations on specific topics
www-01.ibm.com/software/info/education/assistant/
- *Corporate Education Licensing Program (CELP)*
Solutions for large IBM customers who need to adopt IBM Tivoli's tools and technologies
ibm.com/training/us/tivoli/celp
- *Tivoli training paths*
Course maps with flow charts and course descriptions to help you find the right course
[ibm.com/training/us/tivoli\(paths](http://ibm.com/training/us/tivoli(paths)



IBM Tivoli NetView for z/OS 6.1: REXX Programming

Student's Training Guide

Course: TZ223 ERC: 1.0

August 2011

© Copyright IBM Corp. 2011. All Rights Reserved.

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

IBM, the IBM logo, and ibm.com are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in this publication to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth, savings or other results.

Printed in Ireland

Table of contents

Unit 1: Introduction to REXX programming with NetView for z/OS

Introduction	1-2
Objectives	1-2
Lesson 1: REXX introduction	1-3
REXX overview	1-4
NetView REXX EXECs	1-5
Storage of NetView EXECs	1-6
Common uses of REXX EXECs	1-7
Calling a REXX EXEC	1-8
REXX basics (1 of 2)	1-9
REXX basics (2 of 2)	1-10
Parsing input parameters	1-11
Parsing input parameters with REXX	1-12
Parsing input parameters with NetView	1-13
Sending messages to operators	1-14
Using the SAY instruction	1-15
REXX example	1-16
NetView REXX example	1-17
Pausing for operator input	1-18
Lesson 2: REXX and NetView	1-19
REXX addressing environment	1-20
Defining EXECs to NetView	1-21
EXECs on an automated operator	1-22
EXECs under the PPT	1-23
EXECs on another NetView	1-25
%INCLUDE with interpreted REXX	1-26
Nested EXECs	1-27
External call example	1-28
NetView add-ons	1-29
REXX functions that NetView provides	1-30
LISTVAR example	1-31
Commonly used REXX built-in functions (1 of 2)	1-32
Commonly used REXX built-in functions (2 of 2)	1-33
Student exercise	1-34
Lesson 3: Variables	1-35
Variables in REXX	1-36
Overview of NetView global variables	1-37
GLOBALV command parameters (1 of 2)	1-38
GLOBALV PUTT PUTC:	1-38
GLOBALV GETT GETC:	1-39
GLOBALV DEFT DEFC:	1-39
GLOBALV SAVEx RESTOREx PURGEEx:	1-39
GLOBALV command parameters (2 of 2)	1-40
GLOBALV AUTOx:	1-40
GLOBALV TRACEx	1-41
Task global variables	1-42
Common global variables	1-43
Global variable notes (1 of 2)	1-44
Global variable notes (2 of 2)	1-45

Table of contents

GLOBALV with stem variables1-46
Displaying global variables1-48
QRYGLOBL example1-49
Setting common global variables1-50
CGED command1-51
Student exercise1-52
Lesson 4: Processing messages in REXX1-53
Overview of REXX to trap messages (1 of 2)1-54
Overview of REXX to trap messages (2 of 2)1-55
Functions that MSGREAD sets1-56
Example: Trapping a single message1-57
Multiline messages1-58
Messages from automation1-59
Subset of functions that automation sets1-60
Student exercise1-61
Lesson 5: Analyzing problems1-62
RXTRACE entry/exit tracing1-63
RXTRACE program tracing1-64
Example: Trace i program1-65
Processing return codes1-66
Processing standard errors1-67
SIGNAL instructions1-68
Example: NOVALUE1-69
Lesson 6: Additional topics1-70
Sending messages to the z/OS system console1-71
REXX environments1-72
Controlling the REXX environment1-73
Loading EXECs in NetView storage1-74
MemStore function1-75
Managing memStore members1-76
Performance1-77
Security1-78
Summary1-79

Unit 1: Introduction to REXX programming with NetView for z/OS

Tivoli software

IBM

Introduction to REXX programming with NetView for z/OS



© 2011 IBM Corp.

Introduction

This unit provides a background in programming REXX EXECs for your NetView® for z/OS® environment. REXX EXECs are key to improving operator productivity and providing automation.

Objectives

Tivoli software

IBM

Objectives

After completing this unit, you should be able to:

- Explain the basics of REXX EXECs in NetView
- Write NetView REXX EXECS to do the following tasks:
 - Issue commands
 - Trap and parse messages
 - Set and retrieve global variables
 - Trace and automate global variables
 - Perform automation

1-2

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Lesson 1: REXX introduction

Tivoli software

IBM

Lesson 1: REXX introduction

- Overview
- REXX and NetView
- REXX basics
- Simple examples

1-3

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This lesson provides a brief introduction to REXX and NetView REXX.

REXX overview

Tivoli software 

REXX overview

- REXX = Restructured Extended Executor language
- Set of commands and instructions that form a program, called an EXEC
For example, IF-THEN
- Interpreted when executed, statement-by-statement
Compile for improved performance
- Stored as a member in a data set
- Structured programming language
- Able to run on multiple platforms

1-4

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

REXX is a structured programming language that is **interpreted** when the program runs. The REXX interpreter operates directly on the program as it runs, line by line and word by word.

Basic REXX EXECs are capable of running on many platforms. One of the operating systems that supports REXX is z/OS. The operating system provides additional commands, environments, or functions in addition to the base that REXX provides. Applications can provide additional commands, environments, or functions. For example, NetView extends the available z/OS REXX functions with some of its own.

NetView REXX EXECs

Tivoli software

IBM

NetView REXX EXECs

- Members of DSICLD concatenation
- NetView supplies many EXECs
- Additional NetView functions
 - Available in NetView environment only
 - For example, common global variables
- For improved performance
 - Compile REXX EXECs
 - Load in NetView storage
 - REXX with PIPEs

1-5

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

NetView REXX EXECs can be created before NetView starts, or while NetView runs. REXX EXECs are saved as members of a partitioned data set (PDS) and are ready for NetView operators to use immediately. Many NetView commands are actually REXX EXECs that call command processors.



Tip: To reduce the time required to read an EXEC from a data set, load the EXEC in NetView storage.

Optionally, REXX EXECs can be compiled to significantly improve performance. The IBM REXX/370 compiler product must be installed on the system where the EXECs are to be compiled. The run-time library must be installed on the system that runs the compiled EXECs.

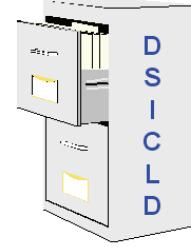
To determine if the REXX compiler is installed, issue ADDRESS LINKMVS 'EAGRTPRQ' within a REXX EXEC. If the return code is -3, it is installed.

Storage of NetView EXECs

Tivoli software **IBM**

Storage of NetView EXECs

- EXECs are data set members
 - Name is eight characters or fewer
 - Data sets are defined in NetView JCL DSICLD concatenation
- Optionally, can be operator data set With OVERRIDE DSICLD= command
- DSICLD concatenation rules apply
 - For example, do **not** define secondary extents



1-6 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Each EXEC is a member of a NetView partitioned data set (PDS), which is allocated to the NetView DSICLD concatenation.

The PDS member name is the EXEC name, the name used by your NetView operators. Optionally, you can define a command synonym name for the EXEC by using the CMDSYN parameter of a command definition. The EXEC name must begin with a nonnumeric character and can be from one to eight characters long. Valid characters are as follows: 0-9 A-Z @ \$ #. Defining EXECs to NetView discussion comes later.

When an EXEC runs, it is read from the partitioned data set. You can load EXECs into storage with the MAPCL and MEMSTORE functions. Discussion of MAPCL and MEMSTORE occurs later.

Avoid using secondary extents when allocating DSICLD data sets. Data sets allocated with secondary extents can create problems for NetView. If the data set goes into secondary extents, you can correct the problem with one of the following actions:

- Issuing the **REACC** command
- Compressing the DSICLD data set by using IEBCOPY in a batch job

You can use the **LISTA** command to list the files that are allocated to the NetView program. Files can be those that are allocated by statements in the JCL and those allocated dynamically by the NetView **ALLOCATE** command.

Common uses of REXX EXECs

Tivoli software

IBM

Common uses of REXX EXECs

- Multiple commands combinable into single program
 - z/OS, NetView, CICS, IMS commands
 - Other REXX EXECs
 - NetView PIPEs
 - And so on
- Simplification of operator keystrokes
- Automation use of REXX
 - Reactive: Respond to an event
 - Proactive: Poll for resource status

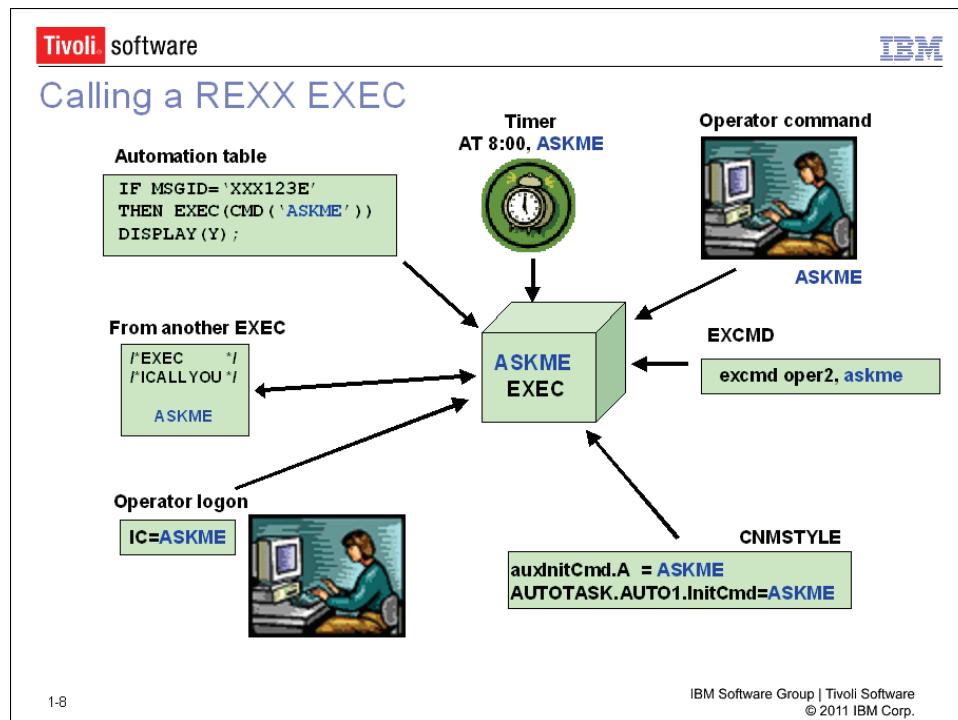
1-7

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Using REXX EXECs, you can combine several commands into one single program, simplifying the number of commands that your operators must issue. EXECs can issue commands, trap and interrogate responses (messages) to the commands, and take more complex actions.

EXECs play a key role in automation. EXECs can be driven directly from an event driving the NetView automation table or scheduled on a timer basis to proactively monitor statuses of resources.

Calling a REXX EXEC



A REXX EXEC can be run in many ways. Calling an EXEC by using some of these methods can force limitations on the commands that can be used within the EXEC. For example, an EXEC that an automated operator runs cannot run full-screen commands.

With an operator station task (OST), an operator can enter commands and receive messages. An OST starts for each operator at logon. This task is used when a NetView operator runs an EXEC. The EXEC can call other EXECs that also run under the OST.

REXX basics (1 of 2)

Tivoli software 

REXX basics (1 of 2)

- First line must be a comment: /* */
- Comments are as follows:
 - Must begin with a /* and end with a */
 - Can start anywhere on a line
 - Can span more than one line (block comment)
- Text strings are as follows:
 - Enclosed within single or double quotation marks
 - 'This is an example of text'
 - "This is another example"
 - Example: SAY "IT'S EIGHT O'CLOCK. TIME TO BRING UP CICS."
- Commands are text strings

1.9 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Each NetView REXX EXEC must begin with a comment. You can also code additional comments in your EXEC wherever necessary.

A series of clauses, each having a separate purpose, make up the functional body of an EXEC. In a simple REXX EXEC, the clauses are interpreted in the sequence that they are coded. The sequence can change by using specific commands that alter the processing order, for example, CALL Label1. (Use quotation marks when you do not want REXX to evaluate the string as a variable.)



Note: Commands are examples of text strings. Place commands within quotations (single or double) so that the REXX interpreter does not perform variable substitution on the string.

Command examples are as follows:

- All NetView commands, including PIPEs
- All MVS commands

REXX basics (2 of 2)

Tivoli software 

REXX basics (2 of 2)

- Variable examples
 - TotalCnt = Cntr1 + Cntr2
 - Stem.i = 5
 - CmdString = 'MYCMD' var1 var2
- Command examples
 - 'MVS D A,L'
 - 'DISPLAY NET,PENDING'
 - 'MYCMD' var1 var2
 - 'MYCMD PARM1='var1' PARM2='var2'
- Continuation example

Say 'You can use a comma',
'to continue to another line.'
- Label examples
 - Parse_and_Validate:
 - Failure:

Commands should be enclosed in quotation marks. Otherwise, REXX treats them as variables.

1-10 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide illustrates examples of variables, commands, continuation, and labels.

'MYCMD' var1 var2 example:

- Suppose variable *var1* has a value of 1 and variable *var2* has a value of 2.
- When MYCMD is called, it would be passed 1 for the first parameter and 2 for the second parameter.
- In this case, MYCMD is the name of a command or another EXEC.

You should enclose the text string, MYCMD, in quotation marks to prevent the REXX interpreter from treating the command name as a variable. If you also had a variable named *mycmd* and omitted quotation marks, the REXX interpreter substitutes the value of the variable as the command name.

Parsing input parameters

Tivoli software **IBM**

Parsing input parameters

- Parse input parameters
 - REXX parse instruction
 - Blank-delimited by default
 - Can change with parse template
 - NetView functions
 - Delimited by blanks, commas, quotation marks, and equal sign
- Use REXX `parse source` to access the command name
 - `parse source . Invoc Command_name .`
- Use REXX parse as an alternative to NetView function calls

1-11 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

EXECs can accept input parameters. The input parameters can be parsed with REXX parsing instructions or NetView functions. REXX parsing is blank-delimited, by default, but can be changed with a parse template that is specified on the **PARSE** instruction.

Using the REXX **PARSE** instruction keeps the EXEC running under the REXX interpreter, which can improve the performance of your REXX EXEC. If you use NetView functions, the REXX interpreter is stopped to call the NetView function.

Use the REXX **parse source** to parse the name of the EXEC (command name). The command name can be used when displaying error messages from the EXEC. REXX also provides a `CmdName()` function to parse the command name.

You can use periods within a parse statement as a placeholder to skip one or more tokens in the text. The parse source example uses periods as follows:

1. Ignore the first token.
2. Place the value of the second token into the *Invoc* variable.
3. Place the value of the third token into the *Command_name* variable.
4. Ignore all remaining tokens.

Parsing input parameters with REXX

Tivoli software 

Parsing input parameters with REXX

- REXX parsing must be done by the EXEC
- Several options
 - **Parse arg argstring or arg argstring or Arg()**
 - Assigns entire input string to argstring variable
Use **parse var** or **parse value** to further parse argstring variable
 - Use **parse upper** if uppercase is required
Simplifies validation of input parameters
 - By default, REXX parse is blank-delimited
Can change by specifying a parse template

1-12

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

REXX parsing is flexible. By default, input parameters are blank-delimited. Suppose an EXEC, MYEXEC, is called with four input parameters: **parm1 parm2 parm3 4,5**

- **parse arg argstring**
Sets the variable, argstring, to the entire input string
- **parse arg p1 p2 p3 p4**
Sets p1 to the text string *parm1*
p2 to the text string *parm2*
p3 to the text string *parm4*
p4 to the text string *4,5*

Arg argstring is a shortened version of a **parse arg argstring** instruction. You can use either format. REXX also provides an Arg() function to parse input parameters.

Parsing input parameters with NetView

Tivoli software

IBM

Parsing input parameters with NetView

- NetView parses input parameters automatically
- You can retrieve with following functions:
 - PARMCNT(): Number of input parameters
 - MSGVAR(n): Value of nth parameter
 - Delimited by blank, comma, single quotation mark, and equal sign
 - Maximum of 31 parameters
 - MSGITEM(n): Current message information delimited by blank or comma
 - MSGCNT(): Number of items in a message string
- Note: Input parameters might be a message from the automation table

1-13

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

When a REXX EXEC is invoked, NetView automatically parses the input to the EXEC and sets the values of several NetView functions.

PARMCNT() identifies the number of input parameters. You can use PARMCNT() to test for the exact number of input parameters and issue a message to the user if an error occurs.

MSGVAR(*n*) can be set to one of the following parameters:

- An input parameter to the EXEC
- A parameter of a message that is read from the trapped message queue

Delimiters that are used for setting these NetView functions are comma, apostrophe, blank, or equal sign.

Sending messages to operators

Tivoli software 

Sending messages to operators

- Use REXX **SAY** instruction to display a single-line message to a NetView operator
- Use NetView **WTO** command to display a single-line message on the system console
- Use NetView **WTOR** command to display a single-line message reply on the system console
 - EXEC waits for response to the reply
 - You take action based on response
- NetView PIPES provide similar functions, plus support for multiline messages

1-14

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can use the REXX **SAY** instruction to send single-line messages to the NetView operator. If a REXX EXEC is issued within a pipeline, any output that the SAY instruction produces becomes input for the next PIPE stage.

You can send single-line messages to the z/OS system console by issuing NetView **WTO** or **WTOR** commands. The **WTOR** command causes the EXEC to wait for a response to the message. The response is contained in the **WTOREPLY** variable.

Using the SAY instruction

Tivoli software **IBM**

Using the SAY instruction

- Write data to screen (maximum of 32,728 characters)
 - Messages
 - Instructions to the NetView operator
- If first token contains a message ID
 - Can satisfy an outstanding WAIT or result in a match in automation table
- Use SAY to describe expected NetView operator input before PARSE EXTERNAL



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The REXX **SAY** instruction is useful for communicating information to the NetView operator running an EXEC. The **SAY** instruction generates a single-line message that drives the NetView automation table.



Note: Be careful when using a message ID in the first token of a SAY instruction. The message ID might generate an event that is trapped erroneously in the automation table, for example. To avoid this, a common practice is to prepend text (for example, three asterisks) to the message ID.

If your EXEC needs to pause for input from the operator, issue a say instruction to describe the required input. Follow the instruction with the **PARSE EXTERNAL** instruction to parse the operator input.

REXX example

The screenshot shows a window titled "REXX example". The window has "Tivoli software" in the top left and the IBM logo in the top right. The main area contains REXX code:

```
/* REXX                                         */
/* SIMPLE EXAMPLE:                           */
/* Display input to EXEC with a SAY          */
/* This EXEC can run in TSO and NetView      */

parse arg p1 .

IF p1= '' THEN           /* no input?      */
  SAY 'Required input parameter missing'
ELSE
  Say 'Input parameter was: ' p1

exit
```

At the bottom left is the number "1-16" and at the bottom right is the text "IBM Software Group | Tivoli Software © 2011 IBM Corp."

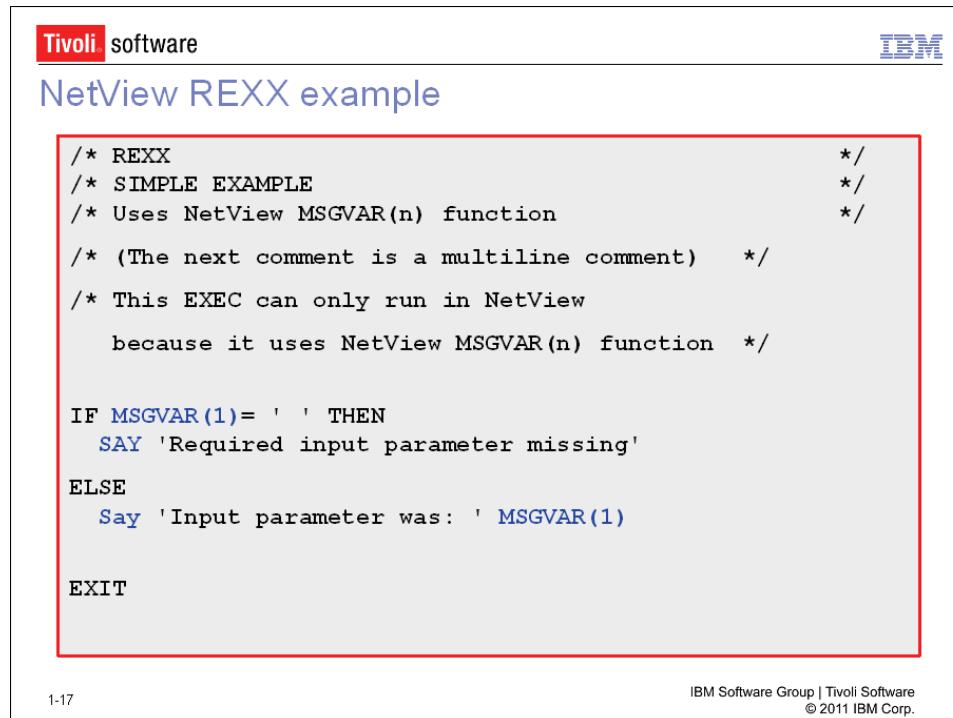
This example REXX EXEC does not use any NetView-specific instructions, functions, or commands. The first four lines are single-line comments. (Note that comments can span multiple lines.)

The **parse arg** instruction parses the input parameters. In this case, the first token is placed into local variable *p1*. All remaining parameters are ignored because of the placement of the period at the end of the statement. The remaining statements in this example EXEC test if variable *p1* is null.

- If *p1* is null, a message routes to the operator: **Required input parameter missing**
- If *p1* is not null, its value is displayed in a message to the operator: **Input parameter was: 5**

Optionally, you can specify a return code on the **exit** instruction. For example, if variable *p1* was null, you can exit with a return code of 8 by coding **exit 8**.

NetView REXX example



The screenshot shows a window titled "NetView REXX example". The window contains REXX code with a red border around the main body. The code is as follows:

```
/* REXX */  
/* SIMPLE EXAMPLE */  
/* Uses NetView MSGVAR(n) function */  
/* (The next comment is a multiline comment) */  
/* This EXEC can only run in NetView  
   because it uses NetView MSGVAR(n) function */  
  
IF MSGVAR(1)= ' ' THEN  
  SAY 'Required input parameter missing'  
ELSE  
  Say 'Input parameter was: ' MSGVAR(1)  
  
EXIT
```

At the bottom left of the window is the number "1-17". At the bottom right is the text "IBM Software Group | Tivoli Software © 2011 IBM Corp."

This is the same REXX example, but using the NetView MSGVAR() function to parse the input to the REXX EXEC. NetView sets MSGVAR(1) automatically to the value of the first input parameter without requiring explicit parsing. Note that lines five and six are a multiline comment.

Pausing for operator input

Tivoli software **IBM**

Pausing for operator input

1. You issue a SAY to request input
`Say 'PLEASE ENTER DATA'`
2. You pause EXEC processing with
`PARSE External p1 p2`
3. "P" is displaced at top right of screen to indicate command is paused
4. EXEC processing stops until operator enters
 - GO command with input
 - CANCEL command
5. PARSE EXTERNAL retrieves data from the command line

`GO MYDATA 123456789`

`PLEASE ENTER DATA`

???

1-18 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can use **PARSE EXTERNAL** or **PARSE PULL** instructions to wait for input from the operator. You should issue a **SAY** command to describe the expected input.

If data exists on the REXX *data stack*, it is read before any operator input from the command line if you use **PARSE PULL** or **PULL**. Use PARSE PULL for mixed-case text and PULL for upper-case text.

To ensure that the data is read from the operator only, use **PARSE EXTERNAL** to place the input in a specified variable. For example, suppose you need an operator to respond with YES or NO before continuing processing within a REXX EXEC. Issue a SAY command with instructions, then a PARSE EXTERNAL immediately after the SAY:

```
SAY 'ENTER "GO YES" OR "GO NO" TO CONTINUE'  
PARSE EXTERNAL ANSWER
```

The local variable, *answer*, should contain YES or NO. To require the input in upper case, code as follows:

```
PARSE UPPER EXTERNAL ANSWER
```

Lesson 2: REXX and NetView

Tivoli software



Lesson 2: REXX and NetView

- REXX environments
- Defining EXECs to NetView
- REXX and NetView tasks
- REXX and NetView functions
- Nesting EXECs

1-19

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

REXX addressing environment

Tivoli software 

REXX addressing environment

TSO

NETVIEW  **MVS**



- The REXX addressing environment is used to process REXX commands
- NetView is the default addressing environment for NetView REXX
- Environment can be changed using **REXX ADDRESS** command
- For example, you switch to MVS environment to read a file and then switch back to NetView environment:
 - **ADDRESS MVS**
 - **EXECIO ...** or **ADDRESS MVS EXECIO ...**
 - **ADDRESS NETVIEW**

1-20 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

REXX EXECs that run in NetView automatically are to run in the NetView environment. Commands can also run in environments such as MVS.

In the NETVIEW addressing environment, the entire command string converts to uppercase characters. If you want to issue a command that contains mixed-case text, change the addressing environment to **NETVASIS**:

```
address netvasis 'WTO This is a mixed case message.'
```

The **WTO** command displays *This is a mixed case message.* on the z/OS system console.

You can use the ADDRESS instruction on a single line or multiple lines. Refer to the TSO/E REXX manuals for more information about the REXX ADDRESS instruction.



Note: With DATA REXX, the only environment that the ADDRESS instruction supports is NETVADATA, the host environment for DATA REXX.

Defining EXECs to NetView

Tivoli software

IBM

Defining EXECs to NetView

- Not necessary to define EXECs
They must exist in a DSICLD data set
- Can define command synonyms
 - CMDDEF.RECOVERY.CMDSYN=RCVR,RV
 - RECOVERY: Real EXEC name
 - RCVR and RV are synonyms

1-21

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

EXECs are available for use by operators immediately after they are created. If the EXEC is in an operator override data set, you must issue an OVERRIDE DSICLD command to refresh the data set.

EXECs can be defined to NetView to create more meaningful synonym names or for security. Use the **CMDDEF** definition statement to define EXECs to NetView. In this case, the command name is RECOVERY. It has two synonyms: RCVR and RV. Operators can call the EXEC by issuing RECOVERY, RCVR, or RV.

Because EXECs can issue commands, many customers require a level of security for controlling the operators who have access to the EXECs. The EXEC must first be defined to NetView with the **CMDDEF** statement. Then corresponding NetView Command Authorization Table (CAT) or Security Access Facility (SAF) definitions must be created to restrict access to the EXEC or its parameters.

For more information about CMDDEF, see the *Tivoli NetView for z/OS Administration Reference*. For more information about security, see the *Tivoli NetView for z/OS Security Reference*.

EXECs on an automated operator

Tivoli software 

EXECs on an automated operator

One of the primary purposes for writing an EXEC is providing automation

- Automation EXECs run on an automated operator
- Routes directly from the automation table
- Routes from another task with EXCMD command

1-22

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Automation is typically performed on an *automated operator*. REXX EXECs can be specified as the automated actions driven from the automation table on the automated operator. Automated operators do not support the following items:

- Full-screen commands
- Keyboard functions, such as PF keys
- AUTOWRAP

EXECs under the PPT

Tivoli software **IBM**

EXECs under the PPT

EXECs can be run under the NetView primary programmed operator interface task (PPT) as follows:

- Automation table
- Routed from another task with EXCMD
 - For example, EXCMD PPT, MVS START TCPIP
- Additional PPT processes:
 - Timers
 - For example, EVERY 24:00:00, PPT, ID=Daily, MONON PU1
 - NetView initialization
 - From CNMSTYLE processing
- Only one PPT task, which is associated with network NetView

1-23 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The primary program operator interface task (PPT) receives the following message types:

- Unassigned, unsolicited messages from VTAM
- Message traffic that are exchanged between the z/OS system console and VTAM.

The PPT opens a (primary) programmed operator interface (POI) to VTAM to receive these messages.

Each NetView program has one PPT. The PPT starts when the NetView program starts. NetView cannot run without the PPT. If automated tasks (autotasks) are not active during initialization, the PPT task can be used to run critical NetView EXECs.



Note: Ensure that the PPT is not overly utilized. Otherwise, NetView (network) performance problems might occur.

The primary purpose of the PPT is handling VTAM unsolicited messages. At network startup, or times of network problems, such messages might be significant. PPT output is displayed on the z/OS system operator console if it is not routed elsewhere.

The PPT does not support the following commands:

- AUTOWRAP
- BGNSESS
- INPUT
- LOGOFF
- ROUTE
- SET
- SUBMIT
- The following REXX instructions under the PPT:
 - FLUSHQ
 - MSGREAD
 - PARSE EXTERNAL
 - PARSE PULL if there is nothing in the REXX data stack
 - PULL if there is nothing in the REXX data stack
 - TRAP
 - WAIT

EXECs on another NetView

Tivoli software **IBM**

EXECs on another NetView

- RMTCMD examples
 - RMTCMD DOMAIN=NETV2,RECOVERY
 - RMTCMD LU=NETV2,OPERID=AUTO1,SHUTD
- Labeled command examples
 - NETV2: RECOVERY
 - NETV2/AUTO1: SHUTD
- Target operator is logged on as a distributed autotask.
Process starts if operator not already logged on

```
graph LR; A[NetView] -- "RMTCMD autotask started" --> B[NetView]
```

1-24 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

EXECs can run in other NetView domains that use the RMTCMD command. The EXEC must be defined in the DSICLD concatenation of the remote NetView domain.

The RMTCMD command routes system, subsystem, and network commands to a remote NetView domain for processing. The responses to these commands are returned to the RMTCMD issuer. To do this, RMTCMD can use an existing task, if it is already active. If the task specified by the OPERID value is not active, RMTCMD processing automatically starts the task. The command specified with RMTCMD is processed by the task. Responses are returned to the RMTCMD issuer.

RMTCMD supports SNA LU 6.2 and TCP/IP session connectivity.

%INCLUDE with interpreted REXX

Tivoli software **IBM**

%INCLUDE with Interpreted REXX

- Create members of the DSICLD data definition with segments of REXX code
- Use the %INCLUDE capability that the NetView program provided
- Code %INCLUDE function on a separate line, followed by the member name you prepared

The diagram illustrates the interpretation of a NetView program named MYPGM1. The program contains the following code:

```
/*%NETVINCL
%INCLUDE A
-----
%INCLUDE B
-----
```

Segment A is represented by a box labeled 'A' containing three horizontal lines. Segment B is represented by a box labeled 'B' containing three horizontal lines. Arrows point from the '%INCLUDE A' and '%INCLUDE B' lines in the program code to their respective segments A and B.

1-25

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

To facilitate code reuse, you can create members of the DSICLD data definition with segments of REXX code and use the %INCLUDE capability that the NetView program provides. To include a prepared code segment, the first line of your program must begin with /*%NETVINCL, followed by comments of your choice.

Code %INCLUDE anywhere in your program on a separate line, followed by the member name that you prepared. The prepared code segment is present when your program is interpreted as if it had been imbedded in your program.



Important: Do not use the INCLUDE function if you intend to compile your REXX program.

Nested EXECs

Tivoli software **IBM**

Nested EXECs

EXECs can call other EXECs

- Directly, by name
Variables not shared
- With the CALL instruction
 - CALL can invoke an external program
 - Behaves like a program subroutine
 - External program can be REXX or assembler
 - Variables can be shared between programs
 - Called program has access to all variables of the caller
 - Caller has access to all variables used by called program
 - Can be controlled with procedure expose instruction
 - CALL can also invoke a subroutine within the EXEC

1.26

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

REXX EXECs can call other REXX EXECs with operands that are passed from the calling EXEC to the nested EXEC as follows:

- Directly using the EXEC name. When the nested EXEC finishes, only a return code routes to the calling EXEC.
- Using the CALL instruction. In this case, data can return in a variable and the return code can be checked.

To pass variables between the calling EXEC and the nested EXEC, you can use NetView global variables, PIPE SAFE, or PIPE INSTORE.

External call example

Tivoli software IBM

External call example

```
/* PROG1 REXX */
.
.
P1 = 1
P2 = 5
Call prog2 p1 p2 'This is p3'
Say 'Total =' Result
.
.
Exit
```

```
/* PROG2 REXX */
PARSE UPPER ARG R1 R2 MSG
.
.
Say 'Parameter 1 =' r1
Say 'Parameter 2 =' r2
Say 'Parameter 3 =' MSG
Sum_Total = r1 + r2
.
.
Return Sum_Total
```

When PROG1 runs, operator detects:

Parameter 1 = 1
Parameter 2 = 5
Parameter 3 = THIS IS P3
Total = 6

PARSE UPPER converts value of third parameter to uppercase.

REXX automatically sets Result variable

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This example shows a use of the CALL instruction. PROG1 calls PROG2 with three parameters:

1. 1
2. 5
3. This is p3

PROG2 parses the input parameters into three variables, displaying them with a SAY instruction. Note that the third parameter is a text string and is also converted to upper case, based on the PARSE UPPER instruction in PROG2.

When an operator runs PROG1, the following information is displayed:

- Parameter 1 = 1 (from the SAY in PROG2)
Parameter 2 = 5 (from the SAY in PROG2)
Parameter 3 = THIS IS P3 (from the SAY in PROG2)
Total = 6 (from the SAY in PROG1)

The CALL instruction supports a maximum of 16 nested EXECs. **SIGNAL ON HALT** detects an error in a nested EXEC and terminates all EXECs. Discussion about SIGNAL ON conditions occurs in more detail later.

NetView add-ons

Tivoli software

IBM

NetView add-ons

- NetView provides many commands and built-in functions for NetView REXX EXECs only (NetView environment).
 - Commands: GLOBALV, TRAP, WAIT, and more
 - Built-in functions: OPID(), SYSID(), JOBNAME(), and more
- See *Programming: REXX and the NetView Command List Language* manual for more information.

1-28

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

NetView provides several instructions to perform NetView specific activities. The NetView-provided instructions can be used only in EXECs that run in a NetView environment. Examples are as follows:

- TRAP: Defines messages that are to be trapped, and specifies whether or not the messages should be displayed to the operator.
- MSGREAD: Causes NetView to read a trapped message from the message queue.
- FLUSHQ: Removes all trapped messages from the message queue that using MSGREAD had not removed.
- WAIT: Temporarily suspends processing of that command procedure until a specified event occurs.
- WAIT CONTINUE: Specifies to continue waiting for additional messages, operator input, or data before command procedure processing resumes.
- GLOBALV: Enables you to define, put, get, automate, and trace global variables in REXX and NetView EXECs.

Discussion about global variables and trapping and processing messages occurs later.

REXX functions that NetView provides

Tivoli software

IBM

REXX functions that NetView provides

- OPID(): Returns the NetView task ID
- DOMAIN(): Returns the NetView domain ID
- NETVIEW(): Returns NetView version and release
- SYSID(): Identifies z/OS system that originated the message
- CURRSYS(): Returns the one to eight character current system name
- MVSLEVEL(): Returns level of z/OS
- TASK(): Returns three character string indicating type of task (PPT, OST, and so on)
- SYSPLEX(): Returns the name of the z/OS sysplex
- TOWER(string): Returns a binary value indicating if a tower or subtower is enabled (1) or disabled (0)
- TOWER(*): Returns the list of enabled towers and subtowers
- MSGID(): ID of a trapped message or a message from the automation table
- ROUTECDE(): Returns the z/OS route codes assigned to the message
- JOBNAME(): Returns the z/OS job name that originated a message

[For more, see Programming: REXX and the NetView Command List Language](#)

1-29

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

NetView provides functions to perform NetView-specific activities. Because these are NetView-specific and not standard REXX functions, you can use them only in EXECs that run in a NetView environment. NetView provides functions as follows:

- Translation
- EXEC information
- Cross-domain information
- Data set information
- Message processing information
- REXX Management Services Units (MSU) information
- Operator information
- Session information
- REXX environment information
- Console information

LISTVAR example

Tivoli software

IBM

LISTVAR example

- LISTVAR uses several NetView REXX functions
- CNM353I LISTVAR : OPSYSTEM = MVS/ESA
- CNM353I LISTVAR : MVSLEVEL = SP7.1.2
- CNM353I LISTVAR : ECVTPSEQ = 01011200
- CNM353I LISTVAR : CURSYS = ADCD1
- CNM353I LISTVAR : VTAMLVL = V61C
- CNM353I LISTVAR : VTCOMPID = 5695-11701-1C0
- CNM353I LISTVAR : NetView = Tivoli NetView for z/OS V6R1
- CNM353I LISTVAR : NETID = ADCD
- CNM353I LISTVAR : DOMAIN = CNM01
- CNM353I LISTVAR : APPLID = CNM01016
- CNM353I LISTVAR : OPID = NETOP1
- CNM353I LISTVAR : LU = SC0TCP22
- CNM353I LISTVAR : TASK = OST
- CNM353I LISTVAR : NCCFCNT = 2
- CNM353I LISTVAR : HCOPY =
- CNM353I LISTVAR : IPV6ENV = MIXED
- CNM353I LISTVAR : TOWERS = NPDA NLDM TCPIPCOLLECT TEMA IPMGT NVSOA DISCOVERY
- CNM353I LISTVAR : CURCONID =

1-30

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The slide shows an example of the NetView **LISTVAR** command. **LISTVAR** displays information that several of the NetView REXX functions contain: APPLID(), OPID(), and others. Issue **BROWSE LISTVAR** to view more details on the LISTVAR command and functions that it uses.

Commonly used REXX built-in functions (1 of 2)

Tivoli software 

Commonly used REXX built-in functions (1 of 2)

- DATE(option): Returns value of date in several formats
 - DATE('U'): mm/dd/yyyy
 - DATE('E'): dd/mm/yyyy
 - DATE('W'): Returns current day of week
- TIME(option): Returns local time
 - TIME('N'): hh:mm:ss
 - TIME('C'): 1:12pm
 - TIME(): 13:12:25
- SUBSTR(string,start,length[,pad]): Returns substring of string that begins at the start position for length characters, right padded with pad character
- STRIP(string[[,option][,char]]): Removes leading, trailing, or both leading and trailing characters (char) from string
Strip(input): Removes leading and trailing blanks from variable input
- POS(needle,haystack[,start]): Returns position of one string, needle, in another, haystack, beginning at start position

1-31 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can use many REXX built-in functions in your NetView REXX EXECs. This slide shows examples of using the DATE() and TIME() built-in functions.

You can manipulate text strings by using the SUBSTR(), STRIP(), and POS() built-in functions. For example, after parsing a text string, remove all leading and trailing blanks from the string by coding as follows:

```
Input = Strip(input)
```

For more information about these and other REXX functions, see *TSO/E REXX Reference Guide*.

Commonly used REXX built-in functions (2 of 2)

Tivoli software

IBM

Commonly used REXX built-in functions (2 of 2)

- COMPARE(string1,string2,pad): Compares two character strings, returns position of first character that does not match
- SYMBOL(variable): Returns VAR if variable is defined, LIT if variable is not
(very important when coding NetView PIPEs)
- WORDS(string): Returns total number of blank-delimited words in string
- WORD(string,n): Returns the nth blank-delimited word in string
- LENGTH(string): Returns the length of string
- Other functions for converting decimal, hexadecimal, and character strings
 - D2X(), X2D()
 - C2X(), X2C()
 - D2C(), C2D()

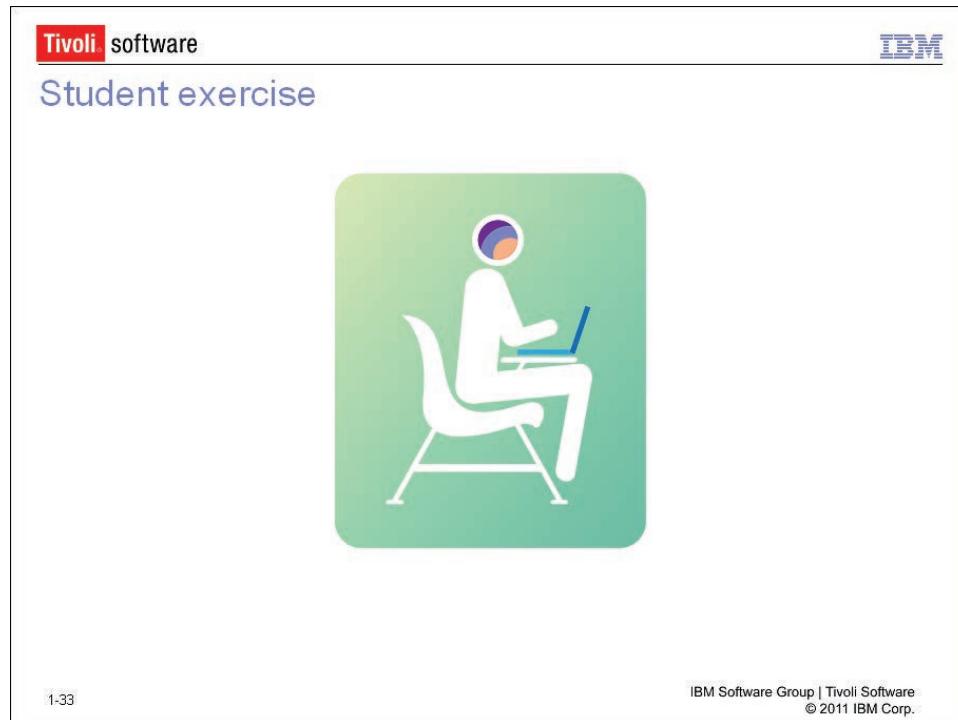
1-32

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide shows more REXX built-in functions that you can use for manipulating text strings. The SYMBOL() function is very useful when using REXX with NetView PIPEs. In some cases, after using a PIPE, you need to test a variable. PIPEs resets variables and, if not set within the pipeline, a NOVALUE condition can occur. To avoid the NOVALUE condition, you can use the SYMBOL() function to test if the variable is set (VAR) or not (LIT). An example is as follows:

```
A = 1
'PIPE NETV command | ... | VAR A'
if SYMBOL(A) = 'LIT' then
    SAY 'Error encountered - variable A is not set'
```

Student exercise



Open your *Student Exercises* book and perform Exercise 1.

Lesson 3: Variables

Tivoli software

IBM

Lesson 3: Variables

- REXX reserved variables
 - **RC** is set by REXX as the return code after every instruction
 - **Result** is set after a CALL instruction
 - **SIGL** is set after a SIGNAL ON condition has been satisfied
 - Your programs should not set these variables
- User variables: Standard REXX
 - Variables available to local EXEC only
 - Also called local variables
 - Can be stem variables
 - Each EXEC has its own set of local variables
- Global variables: NetView specific
 - Variables available to local EXEC, other EXECs on the same task, or other EXECs on other NetView tasks
- **INTERPRET** instruction, usable for dynamically building variable name and value

1-34

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The three types of variables in REXX EXECs are as follows:

- Reserved variables, which REXX sets and that your EXECs should not use.
- User variables, which are the variables that are used within your EXECs.
- Global variables, which are specific to NetView, providing a mechanism for sharing data between EXECs on one task or EXECs on several tasks.

Variables in REXX

Tivoli software

IBM

Variables in REXX

- Mixed case
- Local variables: Name can be one to 255 alphanumeric characters
- Global variables: Name can be one to 31 uppercase alphanumeric characters
- Stem variables supported
- Can be any of the following:
 - A through Z
 - Zero through nine
 - Special characters: . , #, @, \$, _, !, or ?
- First character not to be a number or a period

1-35

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Products from IBM prefix their global variables with a three-character product prefix (for example, CNM or FKX). You can use any other global variable names. If you use a naming convention, you can issue a **QRYGLOBL** command to help display them more easily.

Overview of NetView global variables

Tivoli software **IBM**

Overview of NetView global variables

You can share data between EXECs running on a single task or multiple tasks

- Two types of NetView global variables
 - Task: Shared across EXECs that run on the same task
 - Common: Shared across EXECs that run on all NetView tasks
- **GLOBALV** command to do the following items:
 - Define global variables
Optional, initial value is null
 - Set or retrieve value
 - Save, restore, or purge saved variables
 - Trace global variables
 - Automate on changes to global variables
- **QRYGLOBL** command to display information



1.36 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

NetView provides global variables for use in REXX EXECs.

- Task global variables can be used for sharing data between EXECs that are running on the same task.
- Common global variables can be used for sharing between EXECs that are running on different tasks.

The NetView **GLOBALV** command has several operands to define, set, retrieve, save, restore, trace, automate, and purge saved variables. The NetView **QRYGLOBL** command can display the values of task or common global variables. QRYBALV and QRYGLOBL discussions occur later in this lesson.



Tip: When using GLOBALV in a REXX EXEC, remember to enclose it in single or double quotes.

GLOBALV command parameters (1 of 2)

Tivoli software

IBM

GLOBALV command parameters (1 of 2)

- **PUTT or PUTC:** Store global variable value
Stores value of local (user) variable into NetView global dictionary
- **GETT or GETC:** Get global variable value
- **SAVET or SAVEC:** Save to Save/Restore VSAM file
Wildcard supported (*)
- **DEFT or DEFC:** Defines global variable, does not alter value
 - Defines variable with null value
 - Other EXECs can access the variable after defined
 - Optional
- **RESTORET or RESTOREC:** Restore from Save/Restore VSAM file
Wildcard supported (*)
- **PURGET or PURGEC:** Purge in Save/Restore VSAM file
Wildcard supported (*)

1-37

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

GLOBALV is a NetView command for managing task and common global variables. This section discusses the parameters.

GLOBALV PUTT | PUTC:

Use PUTx to create or update a global variable in the NetView *dictionary* and save its value.



Tip: The local REXX variable must be set to a value before issuing the PUTx. When an EXEC updates the value of the local REXX variable, it should issue another PUTx.

GLOBALV GETT | GETC:

Use GETx to retrieve the value of a global variable. If the global variable is not set, its value becomes null, and the GLOBALV command return code is 144.

```
-----  
V      !  
>>-'GLOBALV --+GETT+---- variable---'-----><  
     '-GETC-'
```

GLOBALV DEFT | DEFC:

Use DEFx to define a global variable in the NetView *dictionary*. It has an initial value of null.

```
-----  
V      !  
>>-'GLOBALV --+DEFT+---- variable---'-----><  
     '-DEFC-'
```

GLOBALV SAVEx | RESTOREx | PURGEx:

Use these options to save, restore, or purge global variables in the NetView Save/Restore VSAM file. SAVEx, RESTOREx, and PURGEx support **wildcards** with an asterisk (*).

```
-----  
V      !  
>>-'GLOBALV --+SAVET+---- variable---'-----><  
     '-SAVEC-'    +- var*-----+  
           '_ *-----'
```

```
-----  
V      !  
>>-'GLOBALV --+RESTORET+---- variable---'-----><  
     '-RESTOREC-'   +- var*-----+  
           '_ *-----'
```

```
-----  
V      !  
>>-'GLOBALV --+PURGET+---- variable---'-----><  
     '-PURGEc-'    +- var*-----+  
           '_ *-----'
```

GLOBALV command parameters (2 of 2)

Tivoli software 

GLOBALV command parameters (2 of 2)

- **AUTOT** or **AUTO_C**: Automate on global variable value
Stores value of local (user) variable into NetView global dictionary
- **TRACET** or **TRACEC**: Trace on global variable value change

1-38 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

GLOBALV AUTO_X:

Use these options to turn on or off automation of global variables. The LIST option displays current automation settings. When a change occurs to a automated variable, a DWO994I message is issued with variable name, old value, and new value of variable. Automation table entries can be coded to process this message. By default, this message is not displayed or logged unless modified by an automation table entry.

```
GLOBALV AUTOT!AUTOC
-----.
. - *-----. V !  
>>-'GLOBALV -+-AUTOT---+----+-----+----+ variable---'-><
     '-AUTOC-' '-OFF-' '- groupID-' +- var-----+
     '- *-----'  
  
GLOBALV AUTOT!AUTOC
-----.
. - *-----.
>>-'GLOBALV -+-AUTOT---+----+-----+-----><
     '-AUTOC-' '-LIST-' +- *ALL----+
     '- groupID-'
```

GLOBALV TRACEEx

Use these options to turn on or off trace of global variable changes. The LIST option displays current traces. When a change occurs to a global variable that is being traced, a DWO990I message is issued with name of the global variable, old value, and new value.

```
GLOBALV TRACET!TRACEC
          .-----.
          . - *-----. V           !
>>-'GLOBALV -+-TRACET-++-ON---+-----+--+ variable-++-'-><
     '-TRACEC-' '-OFF-' '- groupID-' +- var*-----+
          ' - *-----'

GLOBALV TRACET!TRACEC
          .-----.
>>-'GLOBALV -+-TRACET-++-END---+-----+'-----><
     '-TRACEC-' '-LIST-' +- *ALL----+
          '- groupID-'
```

Task global variables

Tivoli software **IBM**

Task global variables

```
/* REXX */
'GLOBALV DEFT VAR1'      /* define task global var1 */
VAR1 = 'abcdefg'          /* update local VAR1 */
'GLOBALV PUTT VAR1'      /* put value into dictionary */

/* VAR1 is available to defining NetView task only */
```

Each task can have its own *local* and *task global* VAR1 with unique values.

1-39 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Task global variables are available to all EXECs running on one task. This slide example shows the following sequence:

1. Defines VAR1 as a task global variable (optional).
2. Sets the value of local REXX variable VAR1 to a character string.
3. Stores the value of local variable VAR1 into the NetView dictionary as task global variable VAR1.



Note: Each time a global variable updates, you should replace its value in the dictionary. For example, if VAR1 updates later in the same EXEC, another **GLOBALV PUTT VAR1** must be issued to save its current value.

Common global variables

Tivoli software **IBM**

Common global variables

```
/* REXX */
'GLOBALV DEFC VAR1'          /* define common global VAR1 */
VAR1 = 'abcdefg'             /* update local VAR1           */
'GLOBALV PUTC VAR1'          /* put value into dictionary */

/* Common global VAR1 is available to any NetView task */
```

Each task can have a *local* VAR1 until it issues a GLOBALV DEFC, GETC, or PUTC for VAR1.

1-40 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Common global variables are available to all EXECs running on all tasks. This slide example shows the following sequence:

1. Defines VAR1 as a common global variable (optional).
2. Sets the value of local REXX variable VAR1 to a character string.
3. Stores the value of local variable VAR1 into the NetView dictionary as common global variable VAR1.

Each time a global variable updates, you should replace its value in the dictionary. For example, if VAR1 updates later in the same EXEC, another **GLOBALV PUTC VAR1** must be issued to save its value. VAR1 can also be updated by another EXEC.

If VAR1 is a counter (for example, counting of the number of times that a resource failed), you should synchronize updates under one task to ensure no loss of data.



Note: You can define VAR1 as a local REXX variable, task global variable, **and** common global variable at the same time.

Global variable notes (1 of 2)

Tivoli software 

Global variable notes (1 of 2)

- The task that is running EXEC maintains local variables in storage
- An internal dictionary that is maintained by NetView stores global variables
Updated with GLOBALV PUTx
- GLOBALV GETx copies value of NetView global variable from the dictionary into local REXX variable of same name
If no global variable exists, null value returned
- GLOBALV PUTx copies local REXX variable to NetView dictionary under global variable of same name

1-41

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Global variables are *snapshots* of local variables that are stored in a dictionary that NetView maintains.



Note: If you put a global variable value to the dictionary and update its value, only the local REXX variable is updated. You must issue another GLOBALV PUTx command to update the dictionary.

Global variable values in the dictionary are not updated dynamically.

Global variable notes (2 of 2)

Tivoli software

IBM

Global variable notes (2 of 2)

- You delete value of NetView global (task or common) variable as follows:
 - Set variable to null (' ') or use REXX **DROP** instruction
 - Issue a **GLOBALV PUTx**
The next **GETx** returns null value
- **PURGE**x deletes from external storage only
- Depending upon use, you might need to serialize common global updates if updates are performed from multiple tasks
For example, counter of failures for a resource



1-42

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

To delete a global variable value, you must set its local variable to null (or use **DROP**). Update the dictionary value with a **GLOBALV PUTx** command. If the global variable is saved to the Save/Restore file, you must issue a **GLOBALV PURGE**x command to remove the global variable from the Save/Restore file.

GLOBALV with stem variables

Tivoli software **IBM**

GLOBALV with stem variables

You must handle PUT and GET stem variables individually, including the stem index variable

```
/* EXEC #1 */
Lcl_Var.0 = 5
Lcl_Var.1 = '1st stem var'
Lcl_Var.2 = '2nd stem var'
Lcl_Var.3 = '3rd stem var'
Lcl_Var.4 = '4th stem var'
Lcl_Var.5 = '5th stem var'

'GLOBALV PUTC LCL_VAR.0 LCL_VAR.1 LCL_VAR.2',
'   LCL_VAR.3 LCL_VAR.4 LCL_VAR.5'
```

```
/* EXEC #2 */
'GLOBALV GETC LCL_VAR.0'

Do i = 1 to LCL_VAR.0
  'GLOBALV GETC LCL_VAR.'i
  Say 'var num' i 'is ...' value('Lcl_Var.'i)
End
```

Issue a GLOBALV **GETC** for **LCL_VAR.0**.
Loop: Issue GLOBALV **GETC** for each stem variable.

Set each stem variable, **Lcl_var.**, plus the stem index, **Lcl_var.0**, and issue a GLOBALV **PUTC** for all six

Put six global variables **Output from second EXEC:**
var num 1 is ... 1st stem var
var num 2 is ... 2nd stem var
var num 3 is ... 3rd stem var
var num 4 is ... 4th stem var
var num 5 is ... 5th stem var

Get six global variables

1-43 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide illustrates special handling of REXX stem variables that are used as global variables. EXEC#1 creates a stem variable, Lcl_Var, with five elements. Each stem variable (Lcl_Var.1 to Lcl_Var.5) is set to a value and put into the common global dictionary. The index (Lcl_Var.0) is also set and put into the common global dictionary.

Before using the stem variable values, EXEC#2 must issue a GLOBALV GETC for each. A typical approach is as follows:

1. Retrieve the stem index (Lcl_Var.0).
2. Create a DO loop within the EXEC to retrieve the *n*th element of the stem variable (Lcl_Var.n).

This set of actions results in putting six global variables in EXEC#1 and retrieving six global variables in EXEC#2.



Note: For maximum performance, minimize the use of GLOBALV GETC commands in EXEC#2 by building a list of common global variables to retrieve. An example follows:

```
VarList = ''  
Do i = 1 to LCL_VAR.0  
    VarList = VarList || Lcl_Var.i  
End  
VarList = Strip(VarList)  
'GLOBALV GETC 'VarList
```

Displaying global variables

Tivoli software **IBM**

Displaying global variables

- Operators can use the **QRYGLOBL** command to display common and task global variables

```
. - BOTH---.
>>-QRYGLOBL--+-+-----+-----+
   +- COMMON+-' - VARS=varspec'
   '- TASK---'

>-+-----+-----+-----><
   '- FILE=membername--+-----+-----+
   '- MODE=modename-'  '- REPLACE-'
```

- Examples
 - QRYGLOBL COMMON VARS=*: Displays all defined common global variables
 - QRYGLOBL TASK VARS=MYVAR*: Displays all defined task global variables that begin with the characters MYVAR
For the issuing task only



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

QRYGLOBL, which either a NetView operator or an EXEC can issue, can be used for displaying NetView common and task global variables.

QRYGLOBL example

Tivoli software

QRYGLOBL example

QRYGLOBL COMMON VARS=CNMSTYLE.AUTO.*
BNH031I NETVIEW GLOBAL VARIABLE INFORMATION
BNH031I COMMAND ISSUED AT: 06/16/11 13:54:03
BNH031I
BNH031I COMMON GLOBAL VARIABLES
BNH031I GLOBAL VARIABLE NAME: GLOBAL VARIABLE VALUE:
BNH031I
BNH039I CNMSTYLE AUTO MVSCLDRREVISION MVSCLDRS
BNH039I CNMSTYLE AUTO SAVEDBMAINT DBAUTO1
BNH039I CNMSTYLE AUTO EXAROTOM AUTO1
BNH039I CNMSTYLE AUTO AUTO1P AUTORON
BNH039I CNMSTYLE AUTO PETS.TCPIP AUTOPPTS
BNH039I CNMSTYLE AUTO TCPS.TCPIP DBAUTO2
BNH039I CNMSTYLE AUTO TCPDBMAINT AUTOTCP5
BNH039I CNMSTYLE AUTO NAPOLTSK2 AUTOC2
BNH039I CNMSTYLE AUTO NAPOLTSK3 AUTOC3
BNH039I CNMSTYLE AUTO NAPOLTSK1 AUTOC1
BNH039I CNMSTYLE AUTO NAPOLTSK4 AUTO4
BNH039I CNMSTYLE AUTO SMONDDBMAINT DBAUTO1
BNH039I CNMSTYLE AUTO OPKT.TCPIP AUTOOPKT
BNH039I CNMSTYLE AUTO COLTSK5 AUTOTCP5
BNH039I CNMSTYLE AUTO XFDISCS AUTOXDSC
BNH039I CNMSTYLE AUTO TCPDBMAINT DBAUTO2
BNH039I CNMSTYLE AUTO MEMSTORE AUTO2
BNH039I CNMSTYLE AUTO ENTDATA AUTOEDAT
BNH039I CNMSTYLE AUTO ISOPAPTSK AUTONVSP
BNH039I CNMSTYLE AUTO IDLEOFF AUTO1
BNH039I CNMSTYLE AUTO NCF AUTONCF
BNH039I CNMSTYLE AUTO COLTSK7 AUTOCT7
BNH039I CNMSTYLE AUTO AP.SERV AUTOTMSI
BNH039I CNMSTYLE AUTO MASTER AUTO1
BNH039I CNMSTYLE AUTO NLCLCLOP AUTONALC
BNH039I CNMSTYLE AUTO POLICY AUTORON
BNH039I CNMSTYLE AUTO TCPCONN.TCPIP AUTOTCP5
BNH039I CNMSTYLE AUTO NETCONV AUTO2
BNH039I CNMSTYLE AUTO DLAAUTO AUTO2
BNH039I CNMSTYLE AUTO COLTSK6 AUTOCT6
BNH039I CNMSTYLE AUTO PREARY AUTO1
BNH039I CNMSTYLE AUTO EXAPTSK AUTOPSRV
BNH035I NUMBER OF VARIABLES FOUND 32
BNH061I
BNH037I NETVIEW GLOBAL VARIABLE INFORMATION COMPLETE

BNH035I contains the number of variables found

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide shows an example of using the QRYGLOBL command to list all of the automation operators (autotasks) defined in CNMSTYLE.

Suppose you want to determine which functions are using AUTO2. You can look for the BNH039I messages and correlate a value of AUTO2 with each global variable name. Or, you could code a NetView PIPE similar to the following one:

```
'PIPE NETV QRYGLOBL COMMON VARS=CNMSTYLE.AUTO.*',
'| CORR',
'| SEP',
'| LOC /BNH039I/ ,
'| LOC / AUTO2/ ,
'| CONS'
```

The results should look similar to examples as follows:

BNH039I CNMSTYLE.AUTO.NETCONV	AUTO2
BNH039I CNMSTYLE.AUTO.MEMSTORE	AUTO2
BNH039I CNMSTYLE.AUTO.DLAAUTO	AUTO2
BNH039I CNMSTYLE.AUTO.MASTER	AUTO2

NetView PIPEs discussion is in the NetView pipelines training module.

Setting common global variables

Tivoli software 

Setting common global variables

- Operators can set common global variables with the **SETCGLOB** and **UPDCGLOB** commands
 - >> SETCGLOB varname TO value -----><
 - >> UPDCGLOB varname -- BY increment -- MAX maxvalue ---><
- Examples
 - SETCGLOB counter TO 101
 - UPDCGLOB counter BY 3 MAX 69
- UPDCGLOB provides serialization of updates
Values are numeric
- Most common use of SETCGLOB and UPDCGLOB come from the automation table



1-46

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

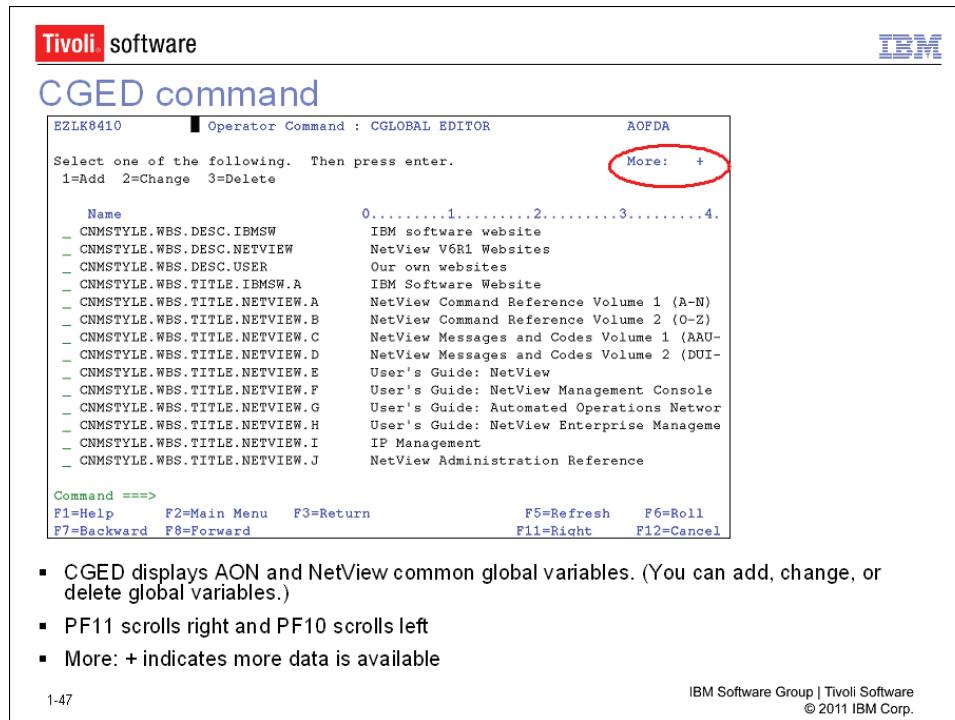
SETCGLOB and **UPDCGLOB** are NetView REXX EXECs for setting common global variables from the automation table. You can use **SETCGLOB** for setting the value of the specified common global variable. Note the following restrictions to this command:

- The use of SETCGLOB is limited to the command procedure and automation environments. (The command must originate in REXX, HLL, automation, or an optional task.)
- The SETCGLOB command sets a common global variable to any value. This command is appropriate to use in any situation where serializing the access among multiple tasks is not important. If serialization of updates is important, use PIPE VARLOAD. If the value is numeric, you can use the UPDCGLOB command.

The **UPDCGLOB** command adds the value that is specified as an increment to the current value of the common global variable. The command runs if the result does not exceed the specified *maxvalue*. If the specified variable is not set (has a null value), it is treated as zero. The new value becomes the value of the increment.

The UPDCGLOB command serializes updates for common global variables by using PIPE VARLOAD to give the effect of a compare and swap logic. If serialization between tasks is not important, you can use the SETCGLOB command instead.

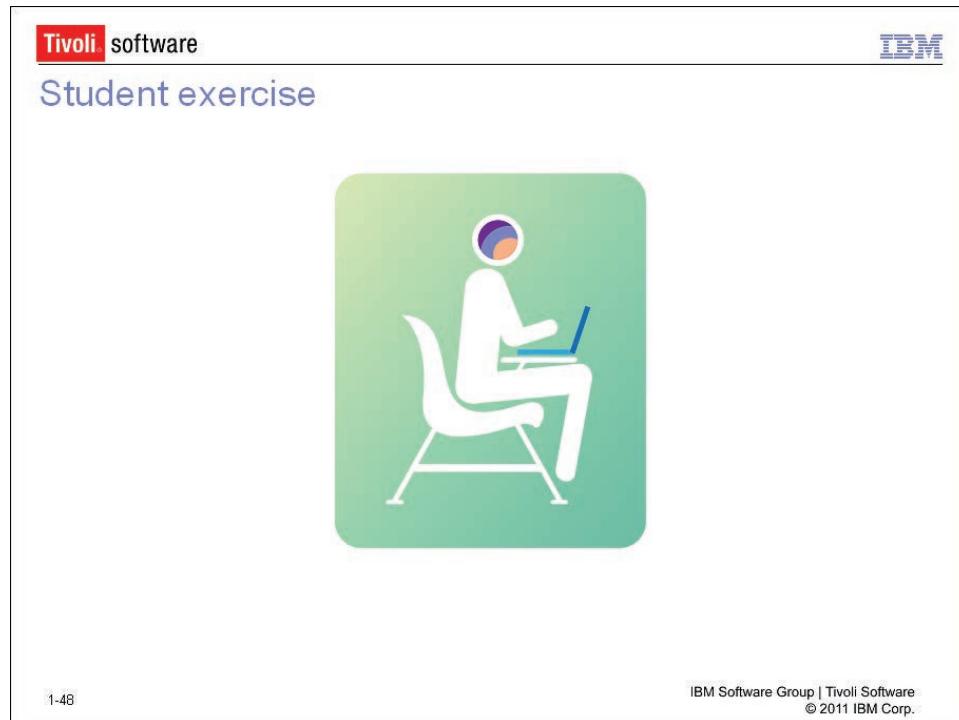
CGED command



CGED displays both AON and NetView common global variables. Options are provided so that you can create new common global variables and delete or modify an existing common global variable. You can also see more information by using your keyboard keys as follows:

- When you see “**More: +**” press PF8 to see more data.
- Press PF10 and PF11 to scroll left and right.

Student exercise



Open your *Student Exercises* book and perform Exercises 2 and 3.

Lesson 4: Processing messages in REXX

Tivoli software

IBM

Lesson 4: Processing messages in REXX

- NetView supplies several instructions to trap and process messages
 - TRAP: Identifies one or more messages to trap
 - Messages must arrive on the task that requests the trap
 - Can be the first line of a multiline message
 - WAIT: Waits to receive messages from a TRAP statement
 - WAIT CONTINUE: Continues to wait for further messages
 - MSGREAD: Reads a message from the trapped message queue
 - FLUSHQ: Purges all messages in the trapped message queue
- Specific to NetView environment only:
Operator (OST), automation operator (AOST), or RMTCMD
- When message is read from trapped message queue, NetView sets several functions, such as MSGID()
- NetView PIPEs provide similar functionality

1-49

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **TRAP** instruction causes NetView to monitor the operator task for specified messages. If the messages occur, they are trapped and added to the *trap message queue*. Trapped messages that fulfil a **WAIT** instruction and can be read by using **MSGREAD**. **MSGREAD** removes the message from the trap message queue and sets several NetView functions with information pertaining to the message. An example is **MSGID()**.

A **WAIT CONTINUE** instruction can be coded to specify that processing should continue to wait for additional messages, operator input, or data before EXEC processing resumes. When processing resumes, the next instruction after the **WAIT CONTINUE** runs.

The **FLUSHQ** instruction is used for removing all trapped messages from the trap message queue that have not been read (removed) using **MSGREAD**.



Note: EXECs that wait for messages result in an error when they run under the PPT task.



Tip: NetView PIPEs provide functions to issue commands and process their responses in a pipeline. *In most cases, you can replace the TRAP, WAIT, and MSGREAD commands (plus the REXX code) with a single pipeline.*

Overview of REXX to trap messages (1 of 2)

Tivoli software 

Overview of REXX to trap messages (1 of 2)

REXX EXECs trapping messages should do the following tasks:

- Issue a TRAP command for defining the messages of interest
 - Both solicited and unsolicited messages can be trapped
 - You must be careful of message assignments and automation table
- Issue the command that is to generate the trapped messages
- Issue a WAIT command to wait for the messages
- Check the type of event that ended the trap
Message is one of the possible event types
- For messages, issue a MSGREAD to access items, such as the message ID and text
- Optionally, code a WAIT CONTINUE to wait for further trapped messages to arrive
- Issue a TRAP NO MESSAGES when finished trapping messages
- Issue a FLUSHQ to purge the trapped message queue to prevent older messages from satisfying a subsequent TRAP

1-50 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can trap a single message, multiple messages, or a multiline message. The basic logic for trapping messages is as follows:

1. Define the messages to trap with a TRAP command.
2. Issue the command that generates the messages being trapped.
3. WAIT for the messages.
During the wait, the operator can see a **W** in the upper corner of the screen.
4. Test the type of event that ended the trap.
Discussion of the four different event types is on the next slide.
5. Use MSGREAD to read the message from the trapped message queue.
6. Continue waiting, if necessary, for more messages.
7. End trapping messages when done to prevent newer messages that would fulfill a message trap.
8. Flush the messages in the trapped message queue.

Messages that occur as a result of a command are *solicited* messages. When specifying a TRAP, be aware that *unsolicited* messages can be trapped. You can test if a message is solicited or unsolicited in your REXX EXECs.

Overview of REXX to trap messages (2 of 2)

Tivoli software

IBM

Overview of REXX to trap messages (2 of 2)

- Check return code set for TRAP, MSGREAD, WAIT commands
- Use NetView **EVENT()** function to check event type that satisfied the wait
 - M: Message received
 - T: Timeout occurred
 - E: Error occurred
 - G: Operator entered a GO command
 - Null: No WAIT was coded
- Use **MSGREAD** to read trapped message
 - Removes message from trapped message queue
 - Sets several NetView functions
 - MSGORIGN(), MSGID(), MSGSTR(), MSGVAR(*n*), SYSID(), and so on
 - Simplifies your REXX coding
- Use REXX to parse the message text: PARSE VAR, SUBSTR(), and so on

1-51

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The REXX-reserved variable, RC, is set after each instruction within an EXEC, including MSGREAD, TRAP, and WAIT instructions.

When a wait is satisfied, you should check the event type that satisfied the wait. In typical operation, the event is message (EVENT()='M'). If the command response does not arrive within the timeout specified on the WAIT instruction, the event becomes a timeout (EVENT()='T').

MSGREAD reads a message from the trap message queue and sets several NetView functions. You can use the NetView functions to parse the message: MSGID(), MSGSTR(), MSGVAR(*n*), and so on. You can also use a REXX PARSE of MSGSTR() to further parse the text of the message.

Functions that MSGREAD sets

Tivoli software

IBM

Functions that MSGREAD sets

- **MSGORIGIN()**: NetView domain where message was generated
- **MSGSTR()**: Text of message, not including message ID
- **MSGID()**: Message ID
- **MSGCNT()**: Count of tokens in MSGSTR()
- **MSGVAR(*n*)**: Message tokens
Maximum of 31 tokens
- Also settable for messages that are routed from the automation table

1-52

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

NetView sets many built-in functions when you issue a MSGREAD command.

Example: Trapping a single message

Tivoli software

IBM

Example: Trapping a single message

```
*****  
/* GREETING - SHOW SIMPLE EXAMPLE OF WAITING AND TRAPPING USING THE DATE COMMAND */  
/* NOTE: WHEN DATE IS ENTERED, THE FOLLOWING IS RETURNED: */  
/* CNM359I DATE : TIME = HH:MM DATE = MM/DD/YY */  
*****  
/*TRAP AND SUPPRESS ONLY MESSAGES CNM359I */ /* TRAP DATE MESSAGE */  
'DATE' /* ISSUE COMMAND */  
'WAIT 10 SECONDS FOR MESSAGES' /* WAIT FOR ANSWER */  
SELECT /* RESULT IS BACK, PROCESS IT... */  
  WHEN (EVENT()='M') THEN /* DID WE GET A MESSAGE? */  
    DO /* Message received */  
      'MSGREAD' /* ... READ IT IN */  
      HOUR=SUBSTR(MSGVAR(5),1,2) /* ... PARSE OUT THE HOUR */  
      SELECT /* GIVE APPROPRIATE GREETING... */  
        WHEN (HOUR<12) THEN /* ...BEFORE NOON? */  
          SAY 'GOOD MORNING'  
        WHEN (HOUR<18) THEN /* ...BEFORE SIX? */  
          SAY 'GOOD AFTERNOON'  
        OTHERWISE /* ...MUST BE NIGHT */  
          SAY 'GOOD EVENING'  
        END /* OF SELECT (APPROPRIATE) */  
    END /* Message received */  
  WHEN (EVENT()='E') THEN /* DID WE GET AN ERROR? */  
    SAY 'ERROR OCCURRED WAITING FOR DATE COMMAND RESPONSE'  
  WHEN (EVENT()='T') THEN /* DID WE GET A TIME-OUT? */  
    SAY 'NO MESSAGE RETURNED FROM DATE COMMAND'  
  OTHERWISE /* OF SELECT (RESULT) */  
END
```

1-53

IBM Software Group | Tivoli Software

© 2011 IBM Corp.

This slide shows an example of a TRAP for one message, CNM359I, that results from issuing the NetView **DATE** command. The message is suppressed. When the message is received, the following actions occur:

1. NetView sets the event function: EVENT()='M'
 2. The EXEC parses the fifth message token.
 3. Based on the value of the fifth token, a greeting message goes out.

This example is also coded to handle errors, EVENT()='E,' and wait timeout conditions, EVENT()='T.'

You can use the following example as a sample for coding many other REXX EXECs to TRAP and WAIT for messages.

1. Modify the TRAP AND SUPPRESS statement.
 2. Issue the command to generate the trapped messages.
 3. Replace the Select-When logic after the MSGREAD command to parse the trapped messages.

Multiline messages

Tivoli software 

Multiline messages

- NetView treats a multiline write to operator (MLWTO) as a single message
- Only the first line satisfies message trap
 - All other message lines are ignored by trap
 - EXEC can access and parse all lines
- MLWTO can contain message ID in each line or in only the first line (header)
- First column (HDRMTYPE) identifies source
 - ' If MLWTO from NetView
 - " If MLWTO not from NetView (for example, z/OS)
 - = If user generated MLWTO

1-54 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

In addition to single-line messages and multiple single-line messages, multiline messages can be trapped and processed. Multiline messages are commonly called multiline write-to-operator (MLWTO) messages.

Only the first line of an MLWTO can be trapped. REXX EXECs can parse any line of the MLWTO. MLWTO messages can be identified by using the HDRMTYPE function. Some MLWTO messages contain a message ID on every line, making the MLWTO appear to be multiple single-line messages.

A method for parsing MLWTO messages is using NetView PIPEs. Discussion about NetView PIPEs is in the NetView Pipelines training module.

Messages from automation

Tivoli software

IBM

Messages from automation

- The automation table can route messages
- You can drive REXX EXECs to issue commands to perform tasks as follows:
 - Collect more detailed data
 - Perform automated actions
 - Perform root cause analysis
 - Update status of failed resources
 - Notify appropriate personnel of the problem
 - More
- Automated messages set many NetView functions that REXX can access

1-55

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

One of the primary purposes of a REXX EXEC is to take action based on events that drive the NetView automation table. Actions can range from collecting more data (pertaining to the event or the resource) to notifying appropriate personnel of the failure. For example, you can use one of the two following notification methods:

- SMTP on z/OS for sending an email
- **System Automation for Integrated Operations Management** (SA IOM) for sending an email or voice notifications

Subset of functions that automation sets

Tivoli software



Subset of functions that automation sets

- **MSGORIGIN()**: NetView domain where message was generated
- **MSGSTR()**: Text of message, not including message ID
- **MSGID()**: Message ID
- **MSGTSTMP()**: Message time stamp in form of hhmmss
- **MSGCNT()**: Count of tokens in MSGSTR()
- **MSGVAR(n)**: Message tokens
Maximum of 31 tokens
- **HDRMTYPE()**: One-character NetView buffer type of the received message
For example, solicited versus unsolicited
- **JOBNAME()**: One- to eight-character z/OS job name identifier
- **JOBNUM()**: Eight-character z/OS job number identifier
- **MCSFLAG()**: Returns of the system message flags in a series of eight on (1) and off (0) EBCDIC characters representing the bits in order
- **AUTOTOKE()**: One- to eight-character name of the MPF automation token
- More

See Programming: REXX and the NetView Command List Language for complete list

1-56

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

When a REXX EXEC is called from the automation table, many functions are set with information that pertain to the event. MSGID() and MSGSTR() have been discussed in relation to trapping messages. They work the same way when called from the automation table. More details about the functions that the automation table processes set are in the *IBM Tivoli NetView for z/OS: Programming: REXX and the NetView Command List Language* manual.

Student exercise

Tivoli software

IBM

Student exercise



1-57

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 4.

Lesson 5: Analyzing problems

Tivoli software

IBM

Lesson 5: Analyzing problems

- Use tracing to aid with debug of EXECs
- Code REXX TRACE instruction within an EXEC
Options: C, E, F, I, L, O, R
- Use RXTRACE command to manage traces for most NetView EXECs
Displays panel to set trace options by operator, NetView domain, or specific EXEC name

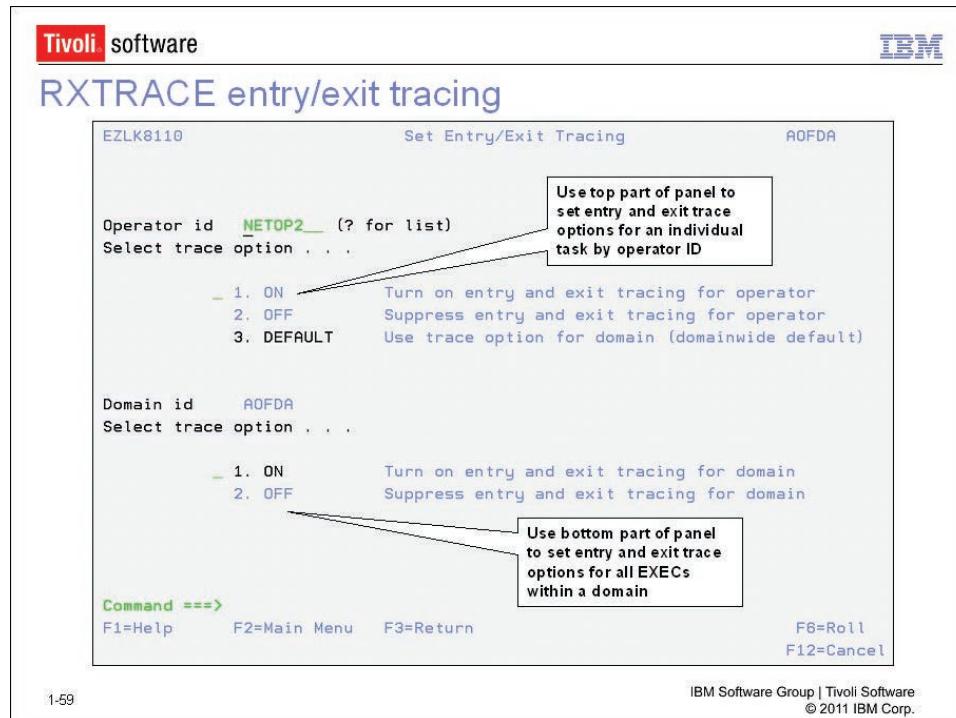
1-58

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Supported REXX TRACE options are as follows:

- **C** (Command): All commands are traced before they run, and any error return code from the commands is displayed.
- **E** (Error): Any command that runs and results in error or failure is traced afterward. Any error return code from the command is also displayed.
- **F** (Failure): Any command that runs and results in failure is traced.
- **I** (Intermediate): All clauses are traced before they run, and any intermediate results are traced during expression evaluation and substitution.
- **L** (Label): All labels that are passed during execution are traced, making it convenient to note all subroutine calls and signals.
- **O** (Off): All trace is turned off, and any previous trace settings are reset.
- **R** (Result): All clauses are traced before they run, and final results of evaluating expressions are traced. This option is useful for general debugging.

RXTRACE entry/exit tracing

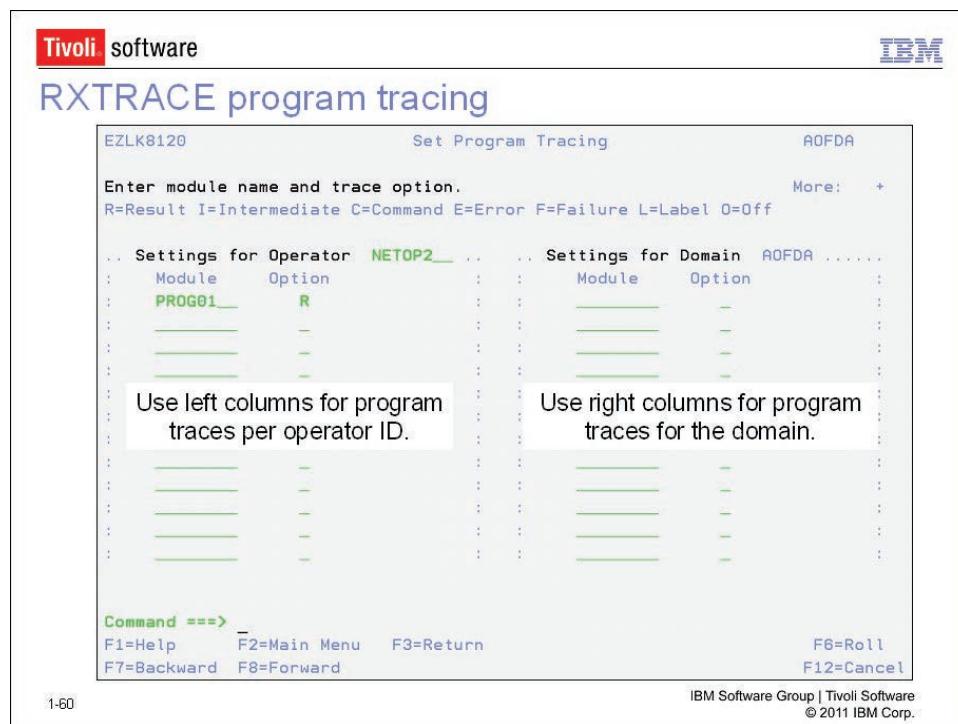


In this slide example, entry/exit tracing is active for the AOFDA domain. NETOP2 is running with the default settings. All supported EXECs create entry and exit trace messages. When entry/exit traces are enabled, EZL260I messages in DSILOG are similar to the following message:

```
EZL260I 09/03/07 17:49:22 NETOP2    EZLE600A ENTRY
EZL260I 09/03/07 17:49:22 NETOP2    EZLE6000 ENTRY
EZL260I 09/03/07 17:49:24 NETOP2    EZLE6000 EXIT      RC=0
EZL260I 09/03/07 17:49:24 NETOP2    EZLE600A EXIT      RC=0
```

- The entry trace identifies the command and input parameters, if any are specified.
- The exit trace identifies the return code for the program.

RXTRACE program tracing



You can enable program traces for many NetView EXECs. You can specify the traces for a single task (for example, NETOP2) or for all tasks in the NetView domain (AOFDA). If you need to trace an EXEC on AUTO2, the following steps apply:

1. Tab over to the *Settings for Operator* field.
2. Enter AUTO2.
3. Press the Enter key.

The EZLK8120 panel displays the current settings for AUTO2. You can add your program trace to the list.

Example: Trace i program

Tivoli software
IBM

Example: Trace i program

```

PROG1
 3 *--* P1 = 1
 >L>   "1"
 4 *--* P2 = 5
 >L>   "5"
 5 *--* Call prog2 p1 p2 'This is p3'
 >V>   "1"
 >V>   "5"
 >O>   "1 5"
 >L>   "This is p3"
 >O>   "1 5 This is p3"

Parameter 1 = 1
Parameter 2 = 5
Parameter 3 = THIS IS P3
    >>>   "6"
 6 *--* If Result = (p1 + p2) ←
 >V>   "6" → If result (value of 6)
 >V>   "1" → p1 (value of 1)
 >V>   "5" → p2 (value of 5)
 >O>   "6" → p1 + p2 (value of 6)
 >O>   "1" → Equals (returns 1 indicating the
             expression evaluated true)
    *--* then
 7 *--* Say 'Total =' Result
 >L>   "Total =" →
 >V>   "6" →
 >O>   "Total = 6" →

Total = 6
10 *--* Exit

```

Trace i:

- Evaluates all clauses, terms, and intermediate results before instruction is run
- Is most comprehensive trace

Line evaluates as follows:

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Processing return codes

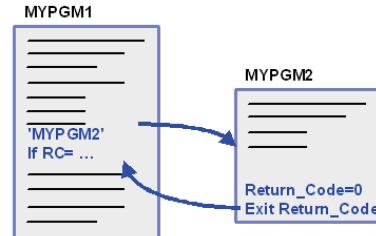
Tivoli software 

Processing return codes

- REXX sets a reserved variable for the return code, RC, after every instruction, command, or nested EXEC
- Example RC values
 - **0:** No error
 - **-1:** Indication of an error
SIGNAL ON FAILURE
 - **-3:** NetView command authorization error
 - **-5:** EXEC canceled
SIGNAL ON HALT
 - **8:** Syntax error
SIGNAL ON SYNTAX
- Other values are possible and depend on NetView component

Do not use RC as a local variable in your EXECs, as *unpredictable results can occur*

1-62 IBM Software Group | Tivoli Software
© 2011 IBM Corp.



Important: RC is one of several variables reserved for REXX use. If you set RC, you create a user variable and nullify the use of the reserved REXX variable.

Processing standard errors

Tivoli software

IBM

Processing standard errors

Use REXX **SIGNAL** instruction to handle error conditions

- Interrupts normal flow, passes control to specified label that matches the signal condition
- Does not return control
- Used for testing and emergency actions
 - For example, save local variables or issue a message about the error
- An example is SIGNAL ON FAILURE
- REXX sets a variable, **SIGL**, to the line number that triggered the condition

1-63

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

SIGNAL instructions

Tivoli software 

SIGNAL instructions



- Code SIGNAL instructions at start of EXEC
 - **SIGNAL ON FAILURE:** Driven for conditions that are not normally recoverable, such as a missing command
 - **SIGNAL ON ERROR:** Driven for recoverable conditions or if failure not coded
 - **SIGNAL ON HALT:** Driven when EXEC is canceled
 - **SIGNAL ON SYNTAX:** Driven when a REXX syntax error is detected
 - **SIGNAL ON NOVALUE:** Driven when an unassigned variable is referenced
- Code label in EXEC to process each following condition:
 - Failure:
 - Error:
 - Halt:
 - Syntax:
 - Novalue:

NetView-supplied EXECs do not use SIGNAL ON ERROR

1-64 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

SIGNAL ON HALT should set return code -5 to cancel all calling EXECs.

Example: NOVALUE

Tivoli software IBM

Example: NOVALUE

```
/*-----*/
/* COMMAND exec FAILED : LINE Sig1 HAD A VARIABLE WITH NO VALUE      */
/*-----*/
NOVALUE:
  Say 'COMMAND' cmd_name 'FAILED : LINE' Sig1 'HAD A VARIABLE WITH NO
  VALUE: ' Condition('D')
  Return_Code = 7          /* return to caller      */
Exit return_code
```

Actions to take when a variable novalue condition is detected

- Branch to the NOVALUE: label
- In this case, issue an error message and exit with a return code 7
 - **Cmd_name**: Command that was parsed within the REXX EXEC with a parse source statement
 - **Sig1**: Line number that the novalue condition sets
 - **Condition('D')**: Function that contains the variable name

1-65 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Lesson 6: Additional topics

Tivoli software

IBM

Lesson 6: Additional topics

- Sending messages to the z/OS system console
- REXX environments
- Loading EXECs into NetView storage
- Performance
- Security

1-66

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Sending messages to the z/OS system console

Tivoli software

IBM

Sending messages to the z/OS system console

- **WTO** (write to operator): Sends single-line message
- **WTOR** (write to operator with reply): Sends single-line message. EXEC is suspended until response is received
- **DOM** (delete operator message): Removes WTO or WTOR message from one or more z/OS consoles if the message is held and highlighted

1-67

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

REXX environments

Tivoli software 

REXX environments

- Each time a REXX EXEC runs in NetView, a REXX environment is built to support the EXEC
 - Nested EXECs share the same environment
 - Environment is reusable
- If two EXECs run, two environments are necessary
- By default, NetView retains up to three environments per task
 - Can improve overall performance
 - Freed when you log off
 - Controllable with **DEFAULTS** and **OVERRIDE** commands
 - Number of environments retained
 - Amount of storage used

1-68

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

If you exceed the number of REXX environments, a CNM416I message is displayed as follows:

CNM416I REXX INTERPRETER ENVIRONMENT INITIALIZATION FAILED FOR TASK
task, RETURN CODE = 20, REASON CODE = 24

A reason for a CNM416I message is that the number of entries in IRXANCHR is insufficient for your NetView environment.

Request that your system administrator increase the number of REXX environments that are defined in IRXANCHR. IRXANCHR is in SYS1.SAMPLIB. Refer to the *TSO/E REXX Reference* for an explanation about increasing the number of entries in IRXANCHR. To determine the number of IRXANCHR entries needed, see the *IBM Tivoli NetView for z/OS Tuning Guide*.



Note: Any EXEC running on your system requires REXX environments, including EXECs from other products or TSO/E EXECs you write.

Controlling the REXX environment

Tivoli software

IBM

Controlling the REXX environment

- Use DEFAULTS or OVERRIDE commands to change settings
- LIST DEFAULTS:
 - REXXSTOR: DEFAULT
 - REXXENV: 3
 - REXXSLMT: 250
 - REXXSTRF: DISABLE

1-69

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

REXXSTOR: Specifies the amount of storage, in 1K increments, to be acquired by REXX environment initialization processing. The default: NetView is to specify that TSO/E REXX use the TSO/E REXX default.

REXXENV: Specifies the number of active and inactive, but initialized, REXX environments to be retained for each operator. The default is **3**.

REXXSLMT: Specifies the amount of storage, in 1K increments, that a REXX environment is allowed to accumulate before being stopped after its current use completes. The default value is **250**.

REXXSTRF: Specifies if the NetView operator can run REXX EXECs that use the REXX STORAGE function. By default, this is disabled.

Loading EXECs in NetView storage

Tivoli software **IBM**

Loading EXECs in NetView storage

- EXECs is loadable in NetView storage to reduce data set read time
- NetView dynamically loads EXECs in storage
 - MEMSTORE statements in CNMSTYLE
 - LIST MEMSTAT=*: Shows information about members loaded in storage with the MEMSTORE function
 - MEMSTOUT: Controls functions of MEMSTORE
- You can preload EXECs in storage
 - LOADCL: Loads EXEC into NetView virtual storage
 - MAPCL: Displays statistics for EXEC loaded in NetView virtual storage
 - DROPCL: Removes EXEC from NetView virtual storage
 - AUTODROP: Drops less frequently used EXECs that are loaded in storage

1-70 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Loading EXECs in NetView storage reduces the allotted time for reading the EXEC from a data set. By default, NetView uses MEMSTORE statements in CNMSTYLE to automatically load the most commonly used EXECs (and other data set members) into storage.

You can also use the LOADCL command to load EXECs in NetView storage. You can schedule the AUTODROP command to monitor the use of the EXECs and drop less-often used EXECs.

MemStore function

Tivoli software 

MemStore function

- Default CNMSTYLE statements
 - function.autotask.memStore = auto2
Autotask, where MEMSTORE runs
 - memStore.stgLimit = 5%
Limit total storage used (% only)
 - memStore.minHits = 5
Minimum hits for caching
 - memStore.frequency = 2
How often to check, in minutes
- Schedule a timer (every two minutes) under AUTO2 to load members that have been used more than 5 times.
(Allocate 5 percent of NetView storage maximum.)

1.71 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

By default, NetView loads members that are used more than five times into storage, up to a maximum of five percent of NetView storage. Members can be excluded from being loaded in storage with the memStore.never statement, as the following example shows.

```
memStore.never =
    DSIPARM.DSIOPF
    DSIPARM.DSIOPFU
    DSILIST.*
    *.USERMEM
```

You can disable the memStore function by setting the maximum storage limit to zero percent as follows:

```
memStore.stgLimit = 0%
```

Managing memStore members

Tivoli software

IBM

Managing memStore members

Use MEMSTOUT to control the members that memStore loads

- Refresh a member
`MEMSTOUT REFRESH ddname.member`
- Unload a member
`MEMSTOUT UNLOAD ddname.member`

1-72

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can use the MEMSTOUT command to delete (unload) or refresh members that are controlled with the memStore function. You can also use MEMSTOUT to update the list of members to exclude from memStore.

REFRESH specifies that members cached by MEMSTORE are removed from the cache, but can be cached again after being read from disk. While UNLOAD specifies that members cached by MEMSTORE are removed from the cache and not cached again.



Note: A common occurrence pertaining to memStore occurs when you initially create and test a REXX EXEC. After several tests, changes you make do not seem to run. This condition can occur because the EXEC loads in storage. Use MEMSTOUT to remove the EXEC from storage.

Performance

Tivoli software

IBM

Performance

- Achieve optimum performance when EXEC remains in REXX interpreter
Issuing non-REXX commands suspends REXX interpreter and waits for command to complete
- Use REXX instructions instead of NetView, for example
 - Use REXX PARSE instead of NetView PARSEL2R or MSGID() and MSGVAR(n)
 - Use REXX PARSE UPPER instead of NetView UPPER command
 - Store multiline messages into REXX stem variable instead of accessing with NetView GETMSIZE and GETMLINE commands
- Issue a single GLOBALV to retrieve or store many (task or common) global variables instead of issuing a GLOBALV per variable
- Preload heavily used EXECs into NetView storage or use MEMSTORE function to dynamically load EXECs
- Consider use of NetView PIPEs when possible

1.73

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide provides several methods for improving the overall performance of your REXX EXECs. Use REXX instructions and functions whenever possible. Doing so can keep your EXEC in the REXX interpreter and improve performance.

Consider the use of NetView PIPEs as follows:

- Use PIPE QSAM to read from and write to files instead of using REXX EXECIO.
- Use PIPE NETV or MVS with LOC, TOSTRING, TAKE, DROP, and so on, instead of TRAP, WAIT, and MSGREAD.

In addition, limit the use of the REXX INTERPRET for dynamically building statements.

Security

Tivoli software 

Security

- Checks EXECs, keywords, and keyword values for proper authorization
Security checks must be explicitly coded
- Requires SAF or NetView command authorization table (CAT) definitions



1-74

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

EXECs, their keywords, and their keyword values can be secured by using two NetView functions:

- AUTHCHK(): Checks keywords and keyword values.
- AUTHCHKX(): Checks command, keywords, and keyword values.

Documentation is in the *IBM Tivoli NetView for z/OS Programming: REXX and the NetView Command List Language* manual

Your system administrator can create entries in the NetView Command Authorization Table (CAT) or a SAF product to authorize users to the EXEC, keywords, and keyword values. For more information, see the *IBM Tivoli NetView for z/OS Security Reference*.

Summary

Tivoli software

IBM

Summary

Now that you have completed this unit, you can perform the following tasks:

- Explain the basics of REXX EXECs in NetView
- Write NetView REXX EXECs to do the following tasks:
 - Issue commands
 - Trap and parse messages
 - Set and retrieve global variables
 - Trace and automate global variables
 - Perform automation

1-75

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

IBM Tivoli certification and training

In today's global business world, enhancing and maintaining skills is essential to keeping pace with rapidly changing technologies. Businesses need to maximize technology potential and employees need to keep up to date with the latest information. Training and professional certification are two powerful solutions.

Certification

There are many reasons for certification:

- You demonstrate value to your customer through increased overall performance with shorter time cycles to deliver applications.
- Technical certifications assist technical professionals to obtain more visibility to potential customers.
- You differentiate your skills and knowledge from other professionals and stand out as the committed technical professional in today's competitive global world.

Online certification paths are available to guide you through the process for achieving certification in many IBM Tivoli areas. See ibm.com/tivoli/education for more information.

Special offer for having taken this course

Now through 31 December 2011: For having completed this course, you are entitled to a 15% discount on your next examination at any Thomson Prometric testing center worldwide. Use this special promotion code when registering online or by telephone to receive the discount: **15CSWR**. (This offer might be withdrawn. Check with the testing center as described later in this section.)

Role-based certification

All IBM certifications are based on job roles. They focus on a job a person must do with a product, not just the product's features and functions. Tivoli Professional Certification uses the following job roles used:

- IBM Certified Advanced Deployment Professional
- IBM Certified Deployment Professional
- IBM Certified Administrator
- IBM Certified Solution Advisor
- IBM Certified Specialist
- IBM Certified Operator

Training

A broad spectrum of courses, delivery options, and tools helps keep your employees up to date with the latest IBM Tivoli information:

- *Instructor-led training (ILT)*
Live interaction with an IBM instructor, hands-on lab exercises, and networking with your peers from other companies
ibm.com/tivoli/education
- *Instructor-led online (ILO)*
All the benefits of ILT, but savings on travel dollars and training costs
ibm.com/training/ilo
- *Self-paced virtual classes (SPVC)*
Interactive and hands-on exercises on your schedule
ibm.com/training/us/spvc
- *Web-based training (WBT)*
Training anywhere, any time, that saves you money and travel
ibm.com/training/us/tivoli/wbt
- *Multimedia library*
Modules supporting new and experienced learners with fully animated multimedia clips, step-by-step audio, and companion text
ibm.com/software/tivoli/education/multimedialibrary
- *IBM Education Assistant*
More specific, granular web-based training with individual presentations on specific topics
www-01.ibm.com/software/info/education/assistant/
- *Corporate Education Licensing Program (CELP)*
Solutions for large IBM customers who need to adopt IBM Tivoli's tools and technologies
ibm.com/training/us/tivoli/celp
- *Tivoli training paths*
Course maps with flow charts and course descriptions to help you find the right course
ibm.com/training/us/tivoli/path



IBM Tivoli NetView for z/OS 6.1: NetView PIPEs

Student's Training Guide

Course: TZ233 ERC: 1.0

September 2011

© Copyright IBM Corp. 2011. All Rights Reserved.

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

IBM, the IBM logo, and ibm.com are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in this publication to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth, savings or other results.

Printed in Ireland

Table of contents

Unit 1: Introduction to NetView® for z/OS PIPEs

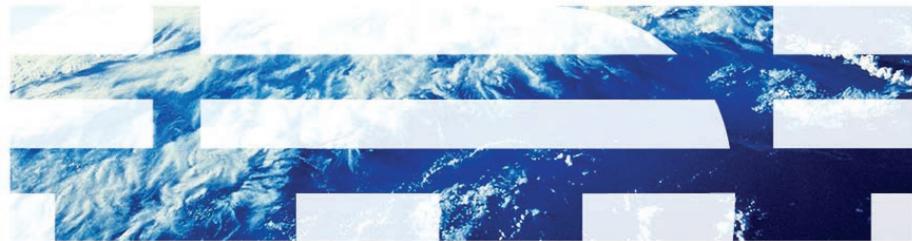
Introduction	1-2
Objectives	1-2
Lesson 1: Overview of NetView pipelines	1-3
Analogy to a water pipeline	1-4
NetView data pipeline	1-5
NetView pipelines overview	1-6
Features of NetView pipelines	1-7
Sources of a NetView PIPE command	1-8
First stage of a PIPE	1-9
First stage of a PIPE: Examples 1 through 3	1-10
First stage of a PIPE: Examples 4 through 6	1-11
Processing PIPE data	1-12
PIPE output	1-13
Simple multiple-stage PIPE	1-14
Asynchronous command responses	1-15
TRAP and WAIT correlation issue	1-16
Correlation with PIPEs	1-17
When correlation is necessary	1-18
Correlation notes	1-19
Lesson 2: PIPE stages	1-20
Simple PIPE command syntax	1-21
Simple example of a complex pipeline	1-22
Interfacing with applications	1-23
Interacting with users	1-24
Working with files	1-25
Storing pipeline data	1-26
Filter stages (1 of 2)	1-27
Filter stages (2 of 2)	1-28
Issuing commands: NETView stage	1-29
Issuing commands: MVS stage	1-30
Simple PIPE example 1	1-31
Simple PIPE example 2	1-32
Simple PIPE example 3	1-33
MOE operand and error messages	1-34
CORRWAIT stage	1-35
CORRCMD stage	1-37
Correlation with PERSIST stage	1-38
Student exercise	1-40
Lesson 3: Using PIPEs to access files	1-41
PIPE QSAM stage	1-42
PIPE < and PIPE > stages	1-42
Security considerations	1-43
PIPE QSAM example	1-44
Example of PIPE <	1-45
MEMLIST stage example	1-46
Student exercise	1-47
Lesson 4: Pipelines and variables	1-48
VAR stage	1-49

PIPE VAR example1-51
STEM stage1-52
Student exercise1-54
Lesson 5: Working with text in pipeline1-55
COLOR Stage: Change message attributes1-56
LOCATE stage: Selecting data from pipeline1-57
NLOCATE stage: Discarding data from pipeline1-58
TOSTRING stage: Ending data stream if string is found1-59
TAKE stage: Selecting data based on line number1-60
DROP stage: Discarding data based on line number1-61
COLLECT and SEPARATE stages: Processing messages1-62
CHOP stage: Truncating pipeline data1-63
BETWEEN stage: Dividing a message stream1-64
REVERSE stage: Reversing order of pipeline data1-65
PICK stage: Selecting specific data1-66
SUBSYM stage: Substituting symbolic variables1-67
CHANGE stage: Replacing string data in pipeline1-68
CASEI stage: comparing strings1-69
SPLIT stage: Dividing data based on string1-70
STRIP stage: Removing characters from a line1-71
SORT stage: Sorting output stream1-72
JOINCONT stage: Combining pipeline records1-73
DUPLICAT stage: Duplicating pipeline records1-74
DELDUPES stage: Removing duplicate records1-75
COUNT stage: Counting numbers of records1-76
Lesson 6: Processing pipeline data1-78
HOLE stage: Deleting all data from pipeline1-79
LOGTO stage: Log messages1-80
SAFE stage: Storing pipeline data1-81
Default SAFE1-81
Named SAFE1-82
SAFE stage examples1-83
KEEP stage1-84
KEEP example1-85
Student exercise1-86
Lesson 7: Advanced topics1-87
Cross-domain pipelines1-88
Pipe within a pipe1-89
Complex pipeline structure1-90
Complex PIPE example1-91
NOT stage: Exchanging input and output streams1-92
Multiple input and output streams1-93
FANIN, FANINANY, and FANOUT1-94
FANIN stage example1-95
Student exercise1-96
Lesson 8: PIPE EDIT stage1-97
PIPE EDIT stage example1-99
PIPE EXPOSE stage1-101
Student exercise1-103
Lesson 9: Full-screen automation1-104
Full-screen automation with a VOST1-105
Student exercise1-106
Summary1-107

Unit 1: Introduction to NetView® for z/OS PIPEs

IBM

Introduction to NetView® for z/OS PIPEs



© 2011 IBM Corp.

Introduction

This unit provides basic information of NetView® for z/OS® PIPEs and the most commonly used PIPE stages. You learn to use PIPE to perform tasks as listed in the slide.

Objectives



Objectives

When you complete this unit, you can perform the following tasks:

- Discuss the basics of NetView PIPEs
- Use NetView PIPEs to perform the following tasks:
 - Issue commands
 - Trap and parse messages
 - Set and retrieve global variables
 - Read from and write to files
 - Perform automation

Lesson 1: Overview of NetView pipelines



Lesson 1: Overview of NetView pipelines

- PIPEs provides stages to perform the following tasks:
 - Issue commands
 - Process messages
 - Access files: VSAM, QSAM, DB2
 - And more
- Extends function of or reduces complexity of the following capabilities:
 - Message handling
 - Message automation, including full screen
 - Communications with non-NetView environments
- Based on UNIX pipelines

1-3

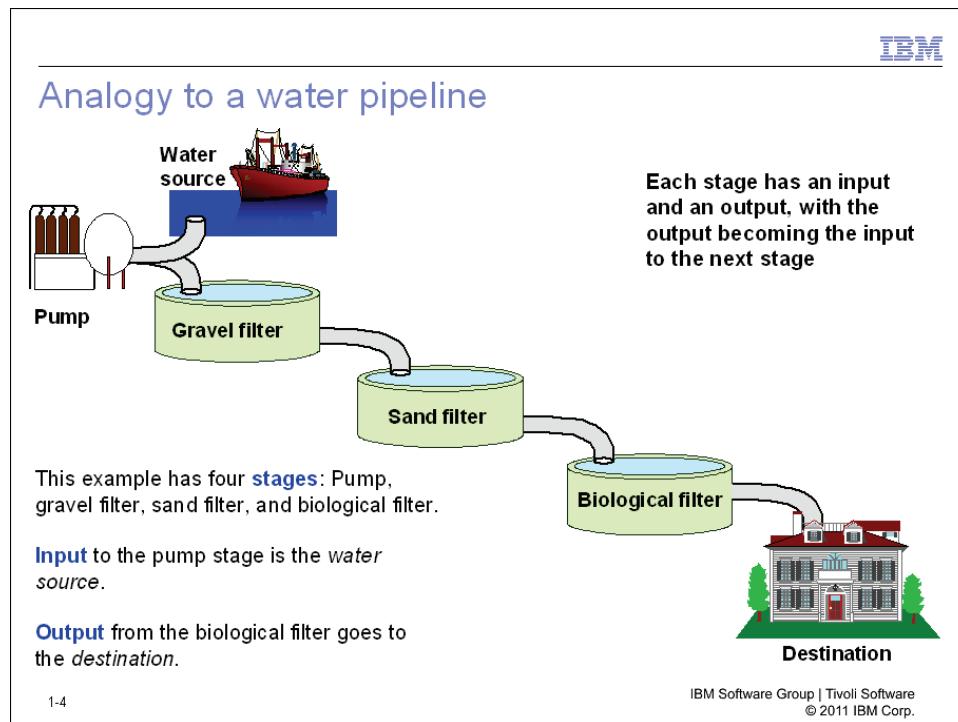
IBM Software Group | Tivoli Software
© 2011 IBM Corp.

NetView pipelines help you solve a complex problem by dividing the problem into a set of smaller, simpler steps. Each step or stage handles one part of the overall problem. PIPE stages can do the following tasks:

- Read data from several sources. For example, it can read files on disk or variables in command procedures.
- Filter and refine the data.
- Export (output) the data from the pipeline.

You can connect stages in logical sequence until they collectively cover all steps that are required to solve your problem.

Analogy to a water pipeline

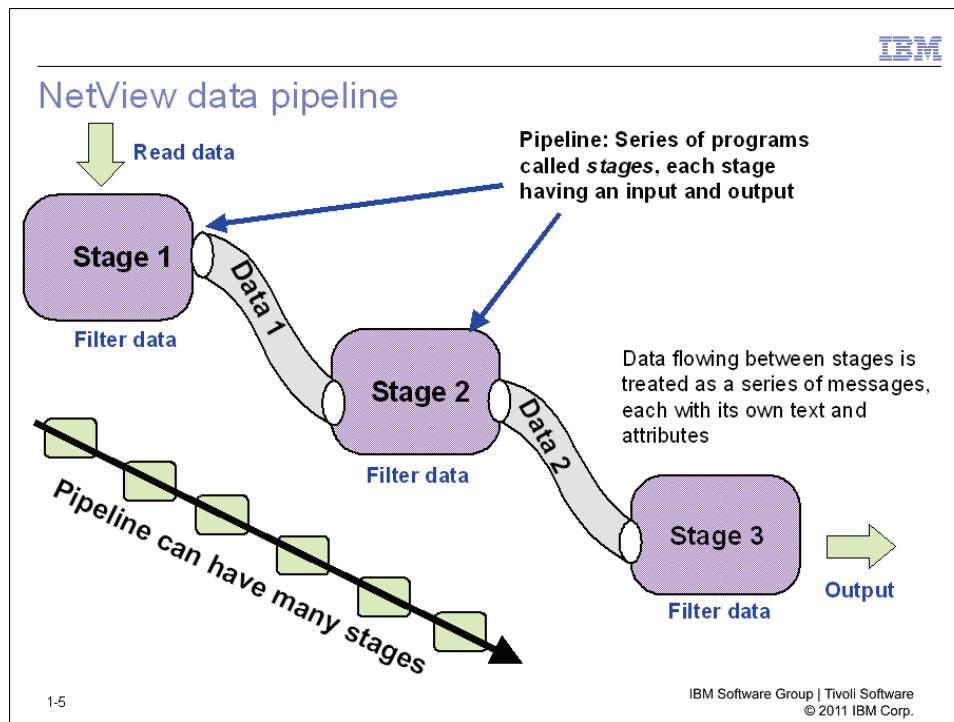


You can understand NetView pipelines if you compare them to a plumbing pipeline in a water treatment process:

Table 1: Comparing plumbing pipeline to NetView pipeline

Plumbing pipeline	NetView pipeline
Receives water from a source. An example is a reservoir or a well.	Receives data from a source. An example is a keyboard or disk.
Passes water through system.	Passes data through stages.
Combines different sizes and shapes of filters to perform complex purification processes.	Combines different stage specifications to perform complex data refinement.
Delivers purified water.	Delivers refined data to other programs or storage.

NetView data pipeline



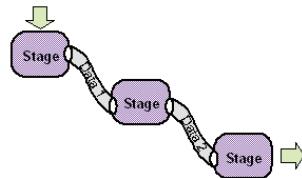
Input data passes through a stage, and a stage command performs an action on it. Several of these stages can be linked together to form a pipeline that produces the output stream that is required.

Each stage in the pipeline acts on the data in the pipeline. For example, stages can edit, delete, add, or display the pipeline data. Each stage has an *input stream* and an *output stream*. When a stage ends, its output stream becomes the input stream for the next stage.

NetView pipelines overview

NetView pipelines overview

- Series of programs: Each program is a **PIPE** stage
- Stages run within NetView **PIPE** command
- Stage separators separate stages
- Data records pass from one stage to the next:
 - Messages include both text and attributes
 - Literal strings
- Stages can modify data records
- Stages can filter data records
- Each stage can act only on data that it receives:
For example, if a previous stage removes a message, subsequent stages do not have access to that message
- You can build complex pipelines



1-6

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Data in the pipeline is a series of discrete records called messages. After it is read into the pipeline, each record becomes a message consisting of message text and message attributes. Data can be discarded, modified, split, ignored, logged, displayed, colored, and so on.

The processing is performed by *pipeline stages*. The *output stream* of one stage is the *input stream* to the next stage. Some stages support two or more input and output streams.

A multiline message is one data record. Multiline messages can be split into multiple single-line data records. Discussions about more complex pipelines (for example, nested and multi-stage pipelines) occur later in this unit.

Features of NetView pipelines



Features of NetView pipelines

- Pipelines are more efficient to code than programs
- Can improve performance
- Uses **correlated responses**, helping to prevent results as follows:
 - REXX, TRAP, and WAIT not correlated
 - Unrelated messages that might fulfill a TRAP condition, causing errors
- Various inputs and outputs
 - Files
 - Messages
 - Console
 - And so on

1-7

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Pipelines are typically more efficient to code and test than programs. Pipeline data results in correlated asynchronous responses.

Sources of a NetView PIPE command



Sources of a NetView PIPE command

REXX EXECs are most common form of pipeline

- Operators
- Automation table
- Command processors

1-8

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Many pipelines start out simple, but can quickly become complicated. Note that there is a difference between a *simple* pipeline and a *complex* pipeline. A complex pipeline has a slightly different structure than the simple type. Discussion of complex pipes occurs later in this unit.

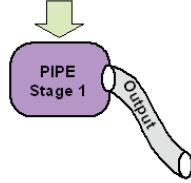
Operators can issue pipelines. REXX EXECs are the most common form of pipelines because the pipeline stages often require programming logic.

First stage of a PIPE

IBM

First stage of a PIPE

- Initial stage generates data records
- Remaining stages process the following tasks or information:
 - Command result
 - Reading records from a file
 - Reading input safe
 - Reading variables: Local or global
 - Writing a text (literal) string
 - Reading contents of a full screen panel
 - Program-to-program interface (PPI)
 - Held-message queue
 - Environment data
 - And more



The diagram shows a purple rounded rectangle labeled "PIPE Stage 1". A green arrow points down to the top of the rectangle. A curved grey arrow labeled "Output" points away from the bottom right corner of the rectangle.

1.9

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The first stage of a pipeline typically issues a command to generate data records for the remaining stages to process. This slide shows several examples of creating data for a pipeline.

First stage of a PIPE: Examples 1 through 3



First stage of a PIPE: Examples 1 through 3

- **Issue NetView command**
PIPE NETVIEW LIST STATUS=TASKS | CORR | ...
- **Issue MVS command**
PIPE MVS D T | TAKE FIRST 1 | ...
- **Read contents of NetView operator profile**
 - PIPE QSAM 'CNM.DSIPRF(DSIPROFB)' | CONS
 - PIPE < DSIPRF.DSIPROFB | CONS

1-10

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Using NetView pipelines, you can perform tasks as follows:

- Issue commands. Examples are NetView and MVS commands.
- Read data from files and write data to files. Pipeline stages are provided to support VSAM, QSAM, and DB2® or SQL files. Discussion of the QSAM and read from (<) pipe stages occur later in this unit.

First stage of a PIPE: Examples 4 through 6



First stage of a PIPE: Examples 4 through 6

- **PIPE STEM myvar. | ...**
Read contents of stem variable, myvar, into pipeline
Note: Stem index variable, Myvar.0, must be set
- **PIPE LITERAL /place this text into the pipeline/ | ...**
 - Write literal string into pipeline
 - Delimiter: First nonblank character that is encountered after LIT stage
 - Use NETVASIC for mixed case
- **pipe var (common) cnmstyle.tcpname |**
Read contents of common global variable named CNMSTYLE.TCPNAME into pipeline

1-11

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide shows examples of generating data records as the first stage of a pipeline. If LITERAL is not the first stage, messages remain in the pipeline, and text added by this stage is inserted in front of the existing messages. You can change the order of pipeline data with the REVERSE or APPEND stages.



Note: You cannot issue STEM and VAR stages from the NetView command line.

Processing PIPE data

Processing PIPE data

- First stage generates data records
- Remaining stages can perform the following tasks:
 - Locate specific data
 - Filter data
 - Split multiline data into single-line
 - Collect multiple single-lines into one multiline
 - Save data into variable
 - Sort data
 - Modify data records
 - Display or log data
 - And so on

Output from stage 1 is input to stage 2,
output from stage 2 is input to stage 3,
and so on

1-12

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The first stage in a pipeline generates data for its output stream to pass to the second stage in the pipeline. The remaining pipeline stages can locate data, sort data, modify data, and so on, passing the data in the output stream for each stage.

Pipeline stages are categorized as follows:

- Device drivers: Transport data but do not take any action on it. Device drivers write their input stream to their output stream. PIPE QSAM and PIPE NETVIEW are examples of device drivers.
- Filters: Examine their input stream, passing data that matches a specified criteria to their *primary output stream*. Data that does not match the specified criteria passes to a *secondary output stream*. PIPE LOCATE and PIPE TAKE are examples of filters.

Pipelines can handle a large amount of data without the need for all of the data to be in storage simultaneously. Some stages (for example, COLLECT) might need to handle all of the data simultaneously.

PIPE output

PIPE output

- NetView operator console
- NetView log
- System log
- Hard copy log
- PIPE SAFE and KEEP
- PIPE HOLE (to discard messages)
- REXX variable (simple or stem)
- File
- PPI
- Command execution

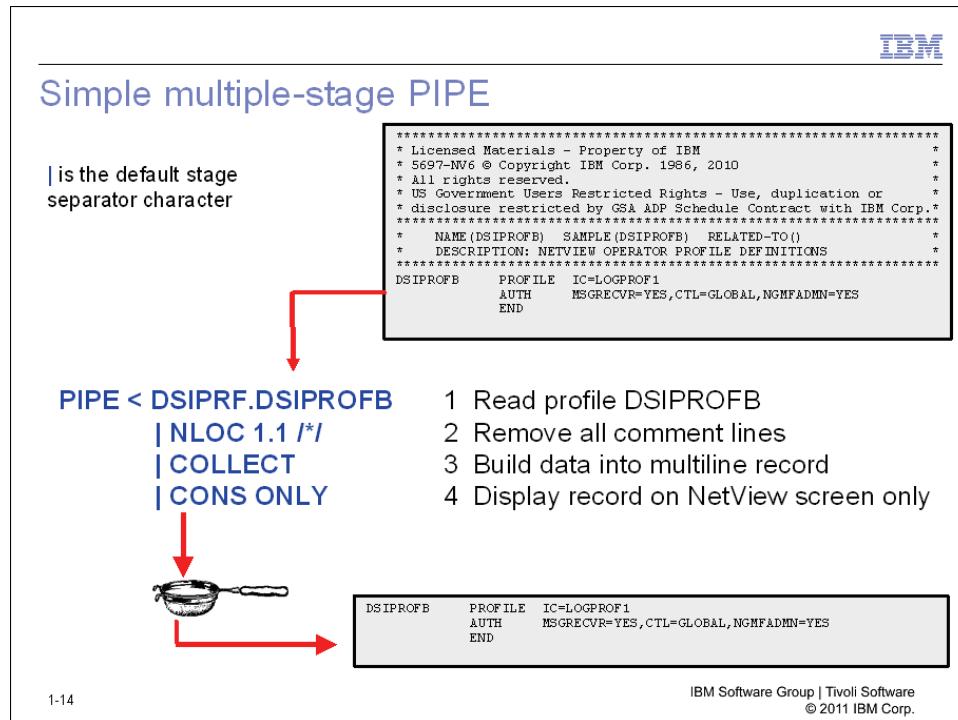
The diagram shows a purple rounded rectangle labeled "PIPE Stage". A curved arrow labeled "Output" points from the stage to a green arrow labeled "Output data".

1-13

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Pipeline data can be displayed, logged, discarded, written to a file, and so on. Note that the slide shows the pipeline output as the last stage. You can code any of the output stages in the middle of a pipeline. Each output stage processes the data in the input stream and passes the data (unmodified) to the output stream. For example, if you are debugging a problem within a pipeline, you can insert the CONS stage to display the contents of the pipeline.

Simple multiple-stage PIPE



This slide shows a simple example of a pipeline with four stages:

1. Read file DSIPROFB in the NetView DSIPRF data set concatenation.
2. Remove any lines that begin with an asterisk (comments).
3. Combine the remaining single-line records into one multiline record.
4. Display the output stream on the NetView screen.

The remainder of this unit discusses each of these PIPE stages and more.

Asynchronous command responses



Asynchronous command responses

- Some commands produce asynchronous output
 For example, MVS commands
- Requires CORRWAIT stage for correlating output
 - PIPE MVS D A,L | CONS results in no output
 - PIPE MVS D A,L | CORR 5 | CONS

```
" AOFDA
IEB1114I 10.11.18 2011.164 ACTIVITY 147
JOBS      M/S   TS USERS   SYSAS   INITS   ACTIVE/MAX VTAM   OAS
00003     00024   00001   00033   00013   00001/00040   00017
CANACN    CANACN  CNDL   NSW   S   LLA   LLA   NSW   S
JES2      JES2    IEFPROC  NSW   S   VLF   VLF   NSW   S
VTAM      VTAM    VTAM   NSW   S   DLF   DLF   NSW   S
RACF      RACF    RACF   NSW   S   TSO   TSO   OWT   S
SDSF      SDSF    SDSF   NSW   S   TCPIP  TCPIP  NSW   S0
TN3270    TN3270  TN3270  NSW   S0  DB9GMSTR  DB9GMSTR  IEFPROC  NSW   S
HTTPD1    HTTPD1  WEBSRV1 IN    SO  NFSS   NFSS   GFSAMAIN NSW   S0
IBMSM    IBMSM   ISM13   NSW   SO  DE9GIRLM DB9GIRLM  NSW   S
DB9GDBM1 DB9GDBM1 IEFPROC  NSW   S  DB9GDISP DB9GDISP  IEFPROC  NSW   S0
PORTMAP  PORTMAP  EMAP   OWT   SO  OSNMPD  OSNMPD  OWT   S0
SNMPQE  SNMPQE   SNMPQE  OWT   SO  FTPD1   STEP1   FTPD   OWT   AO
INETD4   INETD4   OMVSKERN OWT   AO  SSHD4   STEP1   START2  OWT   AO
NETVSSI  NETVSSI  NETVIEW NSW   S   NETVIEW  NETVIEW  NSW   S0
CNMEUNIX CNMEUNIX *OMVSEX  OWT   SO ...
```

IBM Software Group | Tivoli Software
 © 2011 IBM Corp.

Some command responses are *asynchronous* responses. The output of the command does not return to the pipeline without the use of PIPE stages to wait for the response. An example would be a two-stage PIPE that consists of the MVS stage to issue a command and the CONSOLE stage to display the pipeline data. No data would result because of the asynchronous response from the MVS command.

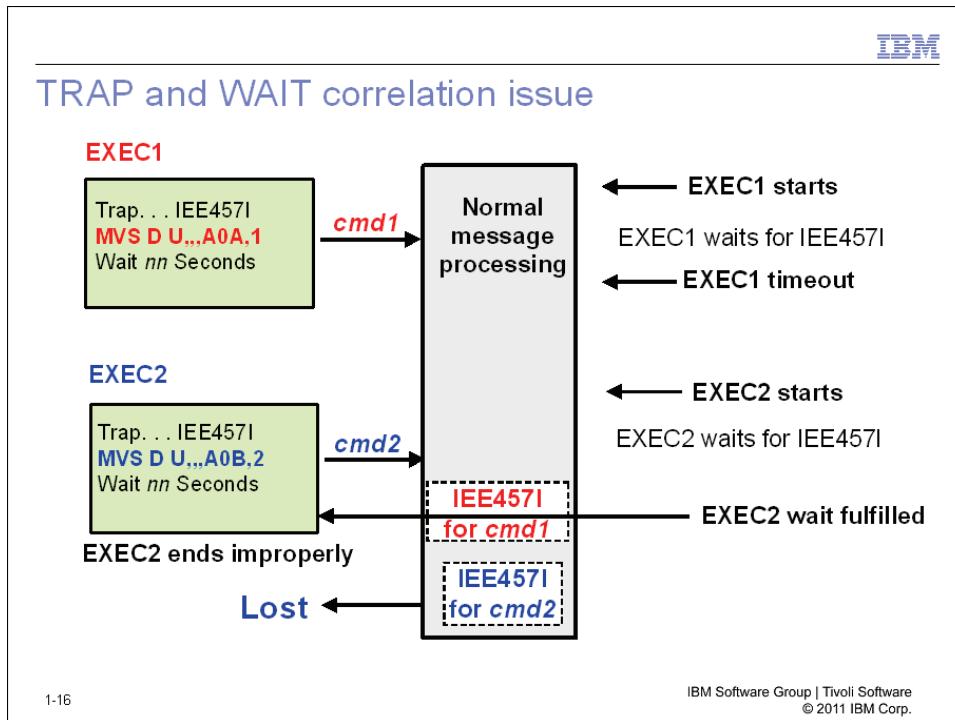
To collect and correlate asynchronous data, the CORRWAIT (WAIT for short) stage must be specified also. It can specify a maximum time to wait for the data



Note: If you do not specify a minimum CORRWAIT time, the PIPE waits one second. If you code CORR *, the PIPE waits forever.

In most cases, you wait for the asynchronous data and specify a *terminating condition* to end the wait as soon as the condition is met. When the PIPE is in a wait state, the task that runs the PIPE also waits. No commands that are queued to the task run until the PIPE completes.

TRAP and WAIT correlation issue



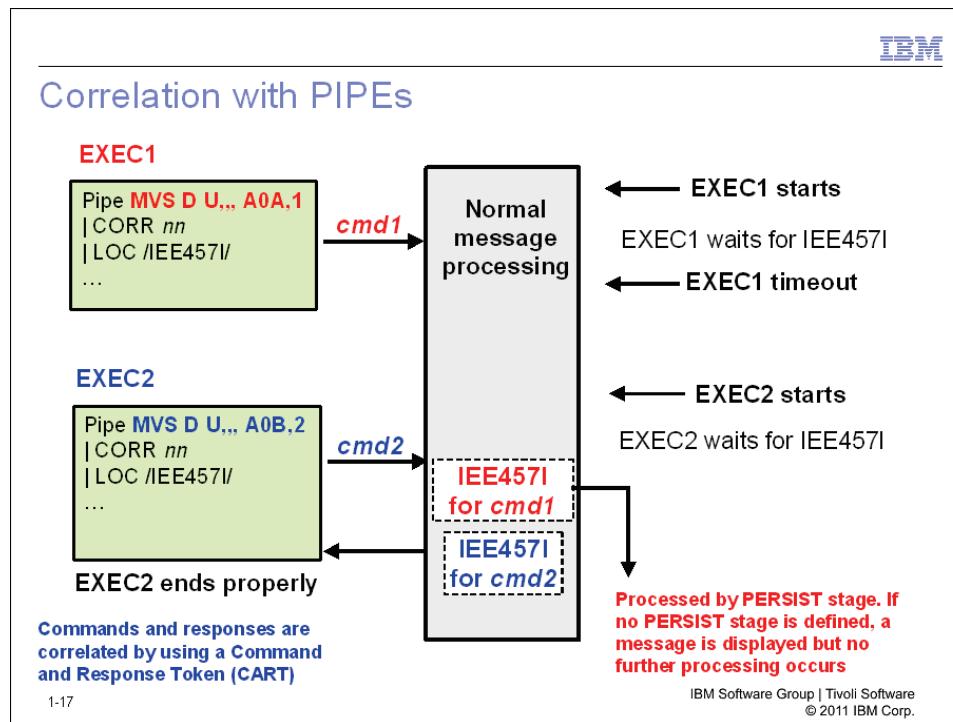
Suppose you have two EXECs that issue the MVS **D U** command. Each EXEC traps the IEE457I message in response to the command. This slide shows a scenario as follows:

1. The first EXEC, EXEC1, encounters a WAIT timeout and ends.
2. EXEC2 is issued and waits for message IEE457I.
3. When the first EE457I message arrives, it is intended for EXEC1.
4. EXEC2 is active and encounters an error because of the erroneous IEE457I message.
5. After EXEC2 ends, the IEE457I that is intended for it arrives and is then discarded.

This scenario is common when writing REXX EXECs to TRAP and WAIT for the same message ID on the same task. Two solutions are as follows:

- You can add code to the EXECs to test the message to make sure it is the correct one.
- You can use a PIPE command similar to the one shown on the next slide.

Correlation with PIPEs



With command correlation, every command that follows the pipeline has a Command and Response Token (CART) appended to it. Only messages that match the CART route to the output stream for a command. This enables the relevant response to be identified and returned asynchronously. Only the correctly correlated replies enter the appropriate pipe.

In the example, the LOCATE stage confirms that each PIPE is waiting for the IEE457I message that pertains to the MVS command issued. The PERSIST stage can process messages that arrive after the PIPE ends. By default, the PERSIST stage displays the message on the task where the PIPE ran. Discussion about PERSIST occurs later in this unit.

Using NetView pipelines resolves two issues as follows:

- All IEE457I messages are correlated and return to the correct pipeline.
- The IEE457I that arrives after EXEC1 completes processes based on the PERSIST stage specification.

When correlation is necessary



When correlation is necessary

- Synchronous responses: Not necessary
Local NetView commands
- Asynchronous responses: Necessary
 - MVS commands
 - VTAM commands
 - UNIX commands
 - TSO commands
 - Commands to remote NetView
- Using CORRWAIT or CORRCMD to wait for responses
- Specifying terminating condition (TOSTRING or TAKE) to end wait
- If no terminating condition, PIPE waits until timeout

1-18

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You should add correlation (CORRWAIT) to your pipelines when issuing the following types of commands:

- Commands to a remote NetView
- Most MVS commands
- VTAM commands
- TSO commands
- UNIX® commands



Note: If you do not specify a minimum CORRWAIT time, the PIPE waits one second. If you code CORR *, the PIPE waits forever.

When the PIPE is in a wait state, the task that runs the PIPE also waits. No commands that are queued to the task run until the PIPE completes.

Correlation notes



Correlation notes

- Not all NetView commands can be correlated
- Some subsystem and MVS commands do not return a correlated response
- If a command is timer-scheduled, output is not trapped in the PIPE that issues the timer command

1-19

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Not all commands are capable of correlation. A subset of older NetView commands are not correlatable. You can test if a command is correlated with a simple PIPE as follows:

```
PIPE NETV command_name | HOLE
```

If you see output for *command_name*, it is not correlated. Correlated command output passes to the HOLE stage and is suppressed. Discussion about the HOLE stage occurs later in this unit.

Lesson 2: PIPE stages



Lesson 2: PIPE stages

- PIPE syntax
- Interfacing with applications
- Interacting with users
- Storing data
- Filtering data
- And more

1-20

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This lesson discusses the PIPE command syntax and several of the most commonly used PIPE stages.

Online help as follows is available when you are logged on to NetView:

- **HELP PIPE SYNTAX:** Displays help for the syntax of the PIPE command.
- **HELP PIPE STAGES:** Displays a list of PIPE stages. Select a stage in the list to see the help.
- **HELP PIPE *stage_name*:** Displays the help for the specified stage. For example, HELP PIPE NETVIEW displays the help for the NETVIEW stage.

Simple PIPE command syntax

IBM

Simple PIPE command syntax

```

>>> PIPE ----->

  .- STAGESEP | ----.
  |-----+-----+-----+-----+
  '- STAGESEP value-'  '- ESC value-'  '- END value-' 

  . | -----
  V   |
>>>-----+-----+-----+-----| Stages |-----><
      '- label: -' '-' (DEBUG) -
  
```

- PIPE can be issued from command line or a procedure
- Many stages supported:
 - Issue commands
 - Read files
 - Parse data records in pipeline
 - And more

1-21

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide shows the basic PIPE command syntax. PIPE stages are separated by a character called a *stage separator*. For example, **PIPE NETVIEW LIST DSMLOG | LOC ... | ... | CONSOLE** uses the default stage separator as follows:

- With the NETVIEW stage to issue a LIST DSMLOG command
- With the LOC stage to filter data
- With the CONSOLE stage to display the results

You can use the **STAGESEP** option to specify the character to use for separating the stages within a pipeline. The default stage separator is the character x'4F'. Depending on the code page used by your workstation, this stage separator is a solid vertical bar (|), split vertical bar (|), or exclamation mark (!).

The **ESC** option indicates that the character that follow the specified character is treated literally when the pipeline specification is parsed. Alternatively, you can use the stage separator character to self-escape itself. Two side-by-side separators resolve to one such character taken literally.

The **END** option identifies a character that is used for signifying the end of a pipeline.

Simple example of a complex pipeline

IBM

Simple example of a complex pipeline

NAMES	
BOB	07/22/2007 ...
TOM	06/01/2007 ...
PHIL	09/06/2007 ...
BOB	08/15/2007 ...
SUE	09/15/2007 ...
.	.
.	.

```
PIPE      (END %)      < NAMES
| A: LOCATE /BOB/
| COLOR RED
| CONSOLE
%A:
| COLOR BLUE
| CONSOLE
```

1. Read data NAMES file
DSIPARM member
2. LOCATE all lines containing string BOB
Display on console in red
3. All other lines: Display in blue

1-22

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide shows a simple example of a *complex pipeline*, a series of simple pipelines that are connected with labels. Some stages accept multiple input streams and others generate multiple output streams. In most cases, when you see an END character that is defined in a pipeline, it is a complex pipeline.

Starting with a simple pipeline, the NAMES file is read (from the NetView DSIPARM data set, by default) with the **PIPE** < stage. The **LOCATE** stage filters out all records that do contain the string BOB and displays them in red on the NetView operator screen.

Using the END character (%), you can specify the end of a pipeline. All output that the first pipeline filters passes to the second pipeline. The NetView operator screen displays the data that is in the second pipeline in blue.

Discussion about complex pipelines occurs later in this unit.

Interfacing with applications

IBM

Interfacing with applications

- **NETView:** Run a specified NetView command
- **MVS:** Run a specified MVS command
- **TSO:** Run a specified TSO command by using the NetView TSO command server
- **UNIX:** Run a specified UNIX command by using the NetView UNIX command server
- **PPI:** Pass data to a PPI receiver queue

Need **CORRWAIT** with MVS, TSO, and UNIX stages

1-23

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Using NetView pipelines, you can issue NetView, MVS, VTAM, TSO, or UNIX commands as follows:

- **NETVIEW stage:** Specifies to run a NetView, MVS, or local VTAM command. An example is PIPE NETV LIST DSMLOG.
- **MVS stage:** Specifies to run an MVS command. The MVS stage can be used instead of coding MVS commands within the NETVIEW stage. An example is PIPE MVS D A,L.
- **TSO stage:** Routes a command to a NetView TSO server, which is a batch job submitted by NetView or a started task.
- **UNIX stage:** Routes a command to a NetView UNIX server where the command is run and the results returned.

You can also use the **PPI** stage to send and receive data from non-NetView applications by using the NetView program-to-program interface (PPI).

Interacting with users

Interacting with users



- **CONsole:** Display contents of pipeline
- **LITeral:** Insert text string into pipeline
- **LOGto:** Log contents of pipeline to DSLOG, system log, or hardcopy log
- **HELDmsg:** Read held-message queue into pipeline

1-24

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide shows the pipeline stages that can be used for interacting with an operator. For example, you can display a message to the operator with the CONSOLE stage.



Note: If you specify CONSOLE ONLY, messages are to be displayed but not held, logged, or exposed to automation. For example, use the ONLY option when sending text to the operator.

Working with files



Working with files

- **QSAM:** Read from (and write to) files
- <: Read data from file into pipeline
- >: Write data from pipeline to file

1-25

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide shows the pipeline stages that can be used for reading from and writing to members of partitioned data sets.

The members can also be loaded in NetView storage for quicker access by programs with the INSTORE stage. The members are then read from storage instead of from the disk when you use the QSAM or < stage. The in-storage members are also viewable when operators use the BROWSE command to view a member.

Storing pipeline data

IBM

Storing pipeline data

- **SAFE:** Read data from (and write to) local message queue
 - Default safe name: * (asterisk)
 - Data stored in *named* safes for use later within command group
- **KEEP:** Read data from (and write to) task-wide storage
 - Data stored in *named* keeps for use by other procedures
- **STEM:** Read from (and write to) stem variables
- **VAR:** Read from (and write to) variables
 - Local, task global, and common global

1-26

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide shows the pipeline stages that can be used for storing data. Data can be stored internally in *safes* or *keeps* or in variables and arrays.

A common pipeline practice is using REXX EXECs to issue commands (for example, NetView LIST DSLOG). The EXEC filters the pipeline contents, and stores the output stream in a stem variable. The EXEC can loop through the contents of the stem variable and parse to determine the log that was active, for example.



Note: You can use PIPE EDIT to perform the same logic without calling a REXX procedure.



Note: You cannot issue these stages from the NetView command line.

Filter stages (1 of 2)



Filter stages (1 of 2)

- Overview
 - Issue command, generating multiline response
 - Separate response into multiple single-line records
 - Filter desired text from response
 - Collect multiple single-line messages into multiline message
 - Pass data to output stream
- Data records can be single-line or multiline
 - **SEParate:** Break multiline message into single-line records
 - **COLLect:** Combine single-line records into multiline message
 Useful when logging or displaying messages

1-27

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Using pipeline filters for working with data, you can accomplish the following tasks:

- Separate multiline messages into multiple single-line messages (SEPARATE)
- Collect multiple single-line messages into a multiline message (COLLECT)

Filter stages (2 of 2)



Filter stages (2 of 2)

- Can select data records based on content
 - **LOCate**: Select records that match specified criteria
 - **NLOCate**: Discard records that match specified criteria
 - **TOString**: End data stream when specified character string found
- Can select record by position
 - **TAKE**: Specify number of records to remain in pipeline
 - **DROP**: Specify number of messages to discard from pipeline
- Can discard pipeline data
HOLE: Discard entire contents of pipeline
(Can be used to test if command can be correlated.)

1-28

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Several stages select pipeline messages to include or discard. They read all messages in the pipeline, but write only those that meet selection criteria to the stage output stream. Using selection filters, you can accomplish the following tasks:

- Keep all messages that contain a match for a specified text string (LOCATE).
- Discard all messages that contain a match for a specified text string (NLOCATE).
- Keep messages up to and including the message that contains a match for a specified text string (TOSTRING).
- Keep the first or last *nn* messages in the pipeline (TAKE).
- Discard the first or last *nn* messages from the pipeline (DROP).
- Empty the pipeline of messages after writing them to a variable or testing a command to see if it is correlated (HOLE).

TOSTRING, TAKE, and DROP stages generate a terminating condition for the CORRWAIT stage.

Issuing commands: NETView stage

IBM

Issuing commands: NetView stage

```
|-- NETVIEW --+-----+----->
'-(+-----+ +-----+ +-----+-)'
'--+CGI--+ ! 'NOPANEL- ! '-MOE-'
' -ECHO -'

>+-----+-----+
' - cmd_text - '
```

Run NetView, local VTAM, or MVS command and place results in pipeline:

- **MOE** (*Message on Error*): Insert DWO369I and return code into pipeline after any resulting messages
- **ECHO**: Write command text to pipeline before command runs
- **CGI**: Specify for HTML output
- **NOPANEL**: Disallow display of full-screen panel. Error message is inserted in pipeline
- **Cmd_text**: Command and parameters

1-29

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **NETVIEW** stage (abbreviated as NETV) runs a NetView, MVS, or local VTAM command. Use the NETV stage to issue VTAM DISPLAY, VARY, and MODIFY commands in a local or remote domain. The resulting messages are placed in the pipeline.

Message on error (MOE) examines the return code from the command. If the return code is not zero, MOE inserts message DWO369I (containing the return code) into the stream after any messages that the command generates. If you do not specify MOE, return codes from commands are ignored.

When NOPANEL is specified, a full-screen panel display is disallowed. If a full-screen panel is encountered, message BNH113W is inserted into the pipeline and the command receives an error code.

You can place the NETVIEW stage anywhere in the pipeline specification. If NETVIEW is not the first stage, the command text is optional. The command runs once for each message that the previous stage delivers. Every time the command runs, the input message becomes the current message during the processing.

The NETVIEW stage does not require an output stream, meaning that NETVIEW can be a last stage. Also, if a stage that follows NETVIEW disconnects, the NETVIEW stage continues to process if it has an input stream.

Asynchronous responses are a subset of NetView command responses. To collect and correlate asynchronous data, the CORRWAIT (WAIT for short) stage must follow.

Issuing commands: MVS stage

IBM

Issuing commands: MVS stage

```
|-- MVS --+-----+-----+  
     '- MOE-'   '- cmd_text -'
```

- Run **MVS** command and place results in pipeline:
 - **MOE**: If the return code is not zero, it inserts message DWO369I containing the return code into the stream after messages that the command might have returned
 - *Cmd_text*: Command and parameters
- Only MVS commands that the extended multiple console support (EMCS) consoles issue are to be correlated
- MVS command responses are asynchronous
Use CORRWAIT stage to collect and correlate responses

1-30

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **MVS** stage specifies an MVS command to run. You can use the MVS stage instead of coding MVS commands within the NETVIEW stage. All rules that apply to an MVS command that is issued by using the NETVIEW stage also apply here. Any restrictions that apply to running an MVS command under PIPE NETVIEW also apply to PIPE MVS.

MVS system commands can be correlated only when issued from extended multiple console support (EMCS) consoles.

MVS command responses are *asynchronous*. To collect and correlate asynchronous data, the CORRWAIT (WAIT for short) stage must follow.

Simple PIPE example 1

IBM

Simple PIPE example 1

Pipe NETV QRYGLOBL COMMON VARS=CNMSTYLE.AUTO.* | CONS ONLY

BNH031I NETVIEW GLOBAL VARIABLE INFORMATION		
BNH103I	COMMAND ISSUED AT: 06/13/11 13:13:07	
BNH061I		
BNH032I	COMMON GLOBAL VARIABLES	
BNH032I	GLOBAL VARIABLE NAME: GLOBAL VARIABLE VALUE:	
BNH064I		
BNH039I	CNMSTYLE AUTO NVSCMDPNSN	NVSCMDPNSN
BNH039I	CNMSTYLE AUTO SAVEDMAIN	DBAUTO01
BNH039I	CNMSTYLE AUTO EMRAUTOM	AUTO01
BNH039I	CNMSTYLE AUTO AUTOOLP	AUTOPON
BNH039I	CNMSTYLE AUTO PATS.TCPPIP	AUTOPKTS
BNH039I	CNMSTYLE AUTO HMONDBMAIN	DBAUTO02
BNH039I	CNMSTYLE AUTO TCP5M.TCPPIP	AUTOTCP5
BNH039I	CNMSTYLE AUTO NAPOLTS2	AUTOTC2
BNH039I	CNMSTYLE AUTO NAPOLTS3	AUTOTC3
BNH039I	CNMSTYLE AUTO NAPOLTS4	AUTOTC4
BNH039I	CNMSTYLE AUTO NAPOLTS5	AUTOTC5
BNH039I	CNMSTYLE AUTO SMONDBMAIN	DBAUTO01
BNH039I	CNMSTYLE AUTO OPKT.TCPPIP	AUTOPKTS
BNH039I	CNMSTYLE AUTO COLTSK5	AUTOTCT5
BNH039I	CNMSTYLE AUTO XCEHISCS	AUTOXHSC
BNH039I	CNMSTYLE AUTO TCPDBMAIN	DBAUTO02
BNH039I	CNMSTYLE AUTO MEMSTORE	AUTO02
BNH039I	CNMSTYLE AUTO ENDBRTA	AUTODEBT
BNH039I	CNMSTYLE AUTO NVSORPTSK	AUTONVSP
BNH039I	CNMSTYLE AUTO IDLEOFF	AUTO01
BNH039I	CNMSTYLE AUTO DLMRPT	AUTOXCE
BNH039I	CNMSTYLE AUTO COLTSK7	AUTOTCT7
BNH039I	CNMSTYLE AUTO AP.SERV	AUTOMSSI
BNH039I	CNMSTYLE AUTO MASTER	AUTO01
BNH039I	CNMSTYLE AUTO NALCLOP	AUTONALC
BNH039I	CNMSTYLE AUTO POLICY	AUTOPON
BNH039I	CNMSTYLE AUTO TCPCONN.TCPPIP	AUTOTCP5C
BNH039I	CNMSTYLE AUTO NETCONN	AUTO02
BNH039I	CNMSTYLE AUTO DLARUTO	AUTO02
BNH039I	CNMSTYLE AUTO COLTSK6	AUTOTCT6
BNH039I	CNMSTYLE AUTO PRIMARY	AUTO01
BNH039I	CNMSTYLE AUTO FKOPKTS	AUTOPSRAV
BNH035I	NUMBER OF VARIABLES FOUND: 32	

1.31

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The first example is a simple pipeline that issues the **QRYGLOBL** command to query all NetView common global variables that begin with the **CNMSTYLE.AUTO.** string. The output is to be displayed to only the console. Data in the pipeline can be filtered with LOC, NLOC, TAKE, DROP, and so on.

Simple PIPE example 2

IBM

Simple PIPE example 2

Pipe MVS D A,L | CORRWAIT 10 | CONS

```
NetView V6R1MD      Tivoli NetView     AOFDA PHOLM    06/13/11 13:20:36 U
* AOFDA    PIPE MVS D A,L | CORRWAIT 10 | CONS
" AOFDA
IEE114I 13,20,36 2011.164 ACTIVITY 461
JOBS   M/S   TS  USERS    SYSAS  INITS   ACTIVE/MAX VTAM   O&S
00003  00024  00001  00033  00013  00001/00040  00017
CANACN CANACN  CNDL  NSW   S  LLA    LLA    NSW   S
JES2   JES2    IEFFPROC  NSW   S  VLF    VLF    NSW   S
VTAM   VTAM    VTAM  NSW   S  DLF    DLF    NSW   S
RACF   RACF    RACF  NSW   S  TSO    TSO    STEP1  OUT  S
SDSF   SDSF    SDSF  NSW   S  TCP/IP TCP/IP  NSW   SO
TN3270 TN3270  TN3270 NSW   SO  DB9GMSTR DB9GMSTR IEFFPROC  NSW   S
HTTPD1 HTTPD1  WEBSRV1 IN    SO  NFSS   NFSS   GFSAMAIN NSW   SO
IBMSM  IBMSM   ISM13  NSW   SO  DB9GIRL1 DB9GIRL1 NSW   S
DB9GDBM1 DB9GDBM1 IEFFPROC  NSW   S  DB9GDIST DB9GDIST IEFFPROC  NSW   SO
PORTMAP PORTMAP  PMAP  OUT   SO  CSNMPD  CSNMPD  CSNMPD  OUT  SO
SNMPQE SNMPQE  SNMPQE OUT   SO  FTPD1  STEP1   FTPD   OUT  AO
INETD4 INETD4  STEP1  OMVSKERN OUT   AO  SSHD4  STEP1   START2  OUT  AO
NETVSS1 NETVSS1 NETVIEW NSW   S  NETVIEW NETVIEW NETVIEW NSW   SO
CNMEUNIX CNMEUNIX *OMVSEX OUT   SO
*LOGON* OUT
```

With no terminating condition, the PIPE waits for the full 10 seconds

Add a terminating condition:

Pipe MVS D A,L | CORRWAIT 10 | **TOSTRING /IEE114I/** | CONS

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This example issues the **D A,L** command by using the **MVS** stage. MVS command responses are *asynchronous* and require the addition of the **CORRWAIT** stage to wait (in this case, 10 seconds) for all responses. The **IEE114I** multiline message is to be displayed as soon as it is received, but you wait for 10 seconds for any additional messages.

To end the wait when the **IEE114I** arrives, use a PIPE stage to generate a *terminating condition*. For example, use **TAKE 1** or **TOSTRING /IEE114I/**. If you do not use the **CORRWAIT** stage, you do not see a response.

Simple PIPE example 3

IBM

Simple PIPE example 3

PIPE (STAGESEP .) MVS D A,L . CORRWAIT 10

- . LITERAL /Command to MVS/
- . COLLECT
- . CONS

```

NetView V6R1MO      Tivoli NetView    AOFDA PHOLM   06/13/11 13:40:35
* AOFDA PIPE (STAGESEP .) MVS D A,L . CORRWAIT 10 . LITERAL /COMMAND TO
  MVS/. COLLECT . COLLECT . CONS
| AOFDA
COMMAND TO MVS
IEE114I 13,40,25 2011,164 ACTIVITY 499
JOBS   M/S   TS  USERS   SYSAS   INITS   ACTIVE/MAX VTAM   OAS
00003  00024  00000  0003   00013  00000/00040  00017
CANACN CANACN  CNDL   NSW S  LL A   LL A   LLA   NSW S
JES2   JES2   IEFFPROC NSW S  VLF   VL F   VL F   NSW S
VTAM   VTAM   VTAM   NSW S  DL F   DL F   DL F   NSW S
RACF   RACF   RACF   NSW S  TSO   TSO   STEP1  OUT  S
SDSF   SDSF   SDSF   NSW S  TCP/IP TCP/IP TCP/IP NSW SO
TM3270 TM3270 TM3270 NSW SO DB9GMSTR DB9GMSTR IEFFPROC NSW S
HTTPD1 HTTPD1 WEBSEV1 IN  SO NFSS NFSS GFSAMAIN NSW SO
IBMSM IBMSM  IBM13  NSW SO DB9GIRLM DB9GIRLM NSW S
DB9GDBM1 DB9GDBM1 IEFFPROC NSW S  DB9GDIST DB9GDBIST IEFFPROC NSW SO
PORTMAP PORTMAP PI MAP OUT  SO CSNMPD CSNMPD OSNMPD OUT SO
SNMPQE SNMPQE SNMPQE OUT  SO FTTP1 STEP1 FTPI  OUT AO
INE1D4 STEP1 OHVSKERN OUT AO SSHD4 STEP1 START2 OUT AO
NETVSSI NETVSSI NETVIEW NSW S  NETVIEW NETVIEW NETVIEW NSW SO
CNMEUNIX CNMEUNIX *OMVSEX OUT  SO

```

→

1-33

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This example issues the **D A,L** command by using the **MVS** stage with a *stage separator* of a period (.). The **CORRWAIT** stage is used for waiting for the asynchronous response.

The **LITERAL** stage inserts a text string into the pipeline when the CORRWAIT stage ends. Collect text string and IEE114I into a multiline message by using the **COLLECT** stage. There is no terminating condition. The output is not displayed until the CORRWAIT ends.

The text string is displayed first. Data that appends in the pipeline output displays as *last in, first out* (LIFO), by default. You can change the order by using the **REVERSE** stage.

MOE operand and error messages

MOE operand and error messages

If you specify **MOE** operand on a PIPE stage, test for message DWO369I

```
/* REXX */
'Pipe MVS (MOE) D A,L',
    '| CORRWAIT 10 | TAKE FIRST 1 | STEM MSGTEXT. '
Parse var msgtext.1 msgid msgstr
Select
    When msgid = 'DWO369I' then          /* Bad RC */
        /* Include code to parse DWO369I message      */
    When msgid = 'IEE114I' then          /* All OK */
        /* Code here to process the IEE114I MLWTO      */
    Otherwise
end
```



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The text of a DWO369I resembles the following text:

```
DWO369I  stagename STAGE (stagenum) HAD RETURN CODE retcode
```

Your procedure can display the DWO369I to users to inform them of the error.

CORRWAIT stage

IBM

CORRWAIT stage

The MVS **DISPLAY TCPIP** command issues two messages: EZAOP411 and EZAOP50I

Example PIPE to suppress the EZAOP411:

```
/* REXX Example */
'Pipe MVS D TCPIP',          /* issue D TCPIP command */
  '| CORR 10',                /* wait max 10 seconds */
  '| TAKE FIRST 1',           /* take first message */
  '| CONSOLE'                 /* display on console */

" AOFDA
EZAOP50I TCPIP STATUS REPORT 510
COUNT      TCPIP NAME    VERSION      STATUS
-----      -----
1          TCPIP        CS V1R12     ACTIVE
*** END TCPIP STATUS REPORT ***
```

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **CORRWAIT** stage allows asynchronous messages (generated by the previous stage) to be returned to the pipeline, and sets a specified time limit if needed. Asynchronous messages are those that return to the NetView program from commands running in another application.: Example commands are MVS, VTAM, TSO, UNIX, and commands running at another NetView. When asynchronous responses are to result from a command issued in a pipe, the next stage command must be CORRWAIT. Without CORRWAIT, the messages are lost.

If a message arrives after the time interval expires, it is discarded. In some instances, a very large time interval might be needed. The wait time can be long between a message arrival and time interval expiration.

A *terminating condition* to end the wait state should follow a CORRWAIT stage. **TAKE** and **TOSTRING** are example stages that end the wait. Each causes the **CORRWAIT** stage to disconnect. CORRWAIT without a *terminating condition* (for example TAKE) waits until the timeout value for all messages.



Note: **LOCATE** does not cause a terminating condition.

In example on the slide, the **MVS DISPLAY TCPIP** command is issued with a CORRWAIT of 10 seconds. The **TAKE FIRST 1** defines a terminating condition that takes control when the first message arrives (if fewer than 10 seconds). Discussion of the TAKE stage occurs later in this unit.

In this example, the second message, EZAOP41I, message is suppressed:

```
EZAOP41I 'DISPLAY TCPIP' COMMAND COMPLETED SUCCESSFULLY
```

CORRWAIT can end under the following conditions:

- Operator issues GO command.
- PIPE issues GO command.
- Match is found for expected condition in a filter stage, for example, TOSTRING or TAKE stages.
- CORRWAIT stage timeout occurs.
- PIPEND stage occurs.
- Secondary input stream disconnects.
- Message is received.

The wait indicator (W) is displayed on the operator screen when CORRWAIT waits for messages.



Note: When the PIPE is in a wait state, the task that runs the PIPE also waits. Commands queued to the task do not run until the PIPE completes.

CORRCMD stage

IBM

CORRCMD stage

```

VARY                                         DSICCDEF
  * VTAM vary command
  CORRWAIT 18
  * PERSIST 120 MINUTES ROUTE AUTHRCVR
  TOSTRING LAST 1.7 /IST314I/ 1.7 /IST093I/ 1.7 /IST061I/ 1.8 /IST1132I/,
    1.7 /IST039I/ 1.7 /IST453I/ 1.7 /DWO369I/ 1.7 /IST073I/ 1.8 /IST1149I/,
    1.7 /IST607I/ 1.8 /IST1133I/ 1.7 /IST455I/ 1.7 /IST072I/,
    1.8 /IST1264I/ 1.8 /IST1149I/ 1.7 /IST455I/ 1.7 /IST607I/ 1.7 /IST105I/

  /* REXX */
  /* PIPE to issue a VTAM VARY NET,ACT command using DSICCDEF      */
  /* instead of coding extensive list of terminating conditions.   */
  'Pipe CORRCMD NETV V NET,ACT,ID=xyz',    /* issue VARY command */
    '| CORRWAIT 18',
    '| TOSTRING LAST 1.7 /IST314I/ ... ',
    '| CONSOLE'                      /* display on console */

```

1.36

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

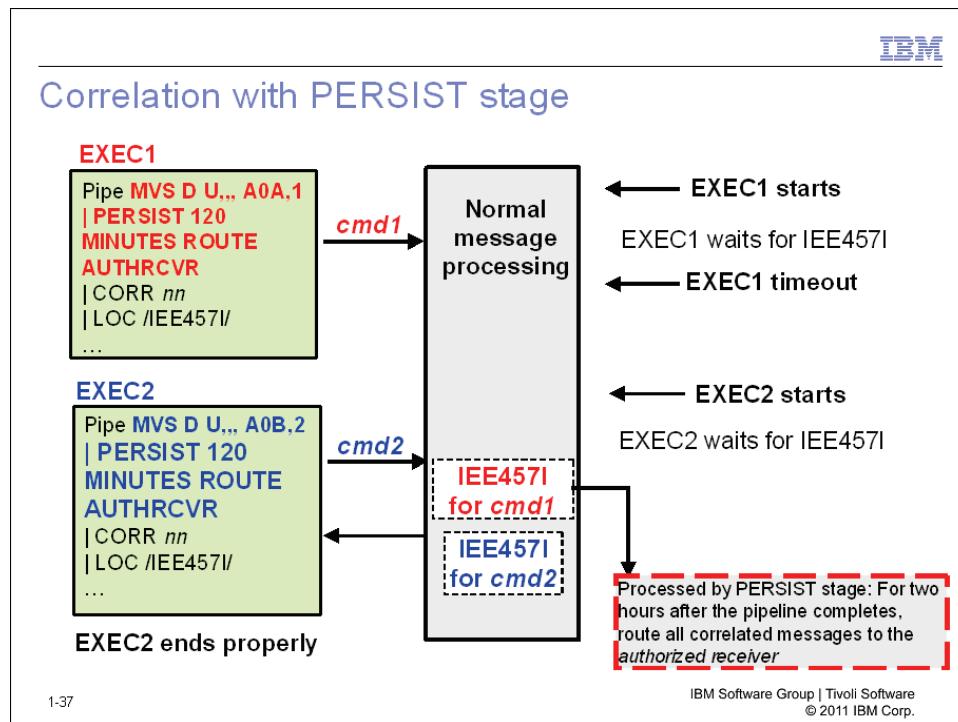
The **CORRCMD** stage (abbreviated as **CC**) processes a command and inserts appropriate correlation wait and termination condition stages. It gathers data from the command being processed. CORRWAIT is included in the inserted stages.

The timeout interval and termination conditions to use are defined in NetView DSIPARM member **DSICCDEF**. A sample file named DSICCDEF (CNMS1082) is provided with NetView. This member contains several commands with appropriate timeout and termination stages. Issue HELP CCDEF or CCDEF QUERY=* NetView commands to display information pertaining to these settings.

This slide shows how to use the sample definitions defined in DSICCDEF for the VTAM VARY command. DSICCDEF defines the CORRWAIT and TOSTRING stages. When a PIPE CORRCMD stage is encountered for a VTAM VARY command, the CORRWAIT and TOSTRING stages append into the pipeline.

To include the PERSIST stage, uncomment the **PERSIST 120 MINUTES ROUTE AUTHRCVR** statement. To simplify the coding of your pipeline stages, use DSICCDEF and CORRCMD

Correlation with PERSIST stage



The **PERSIST** stage defines the action to perform when messages arrive after termination of the correlation that a CORRWAIT stage represents. The conditions that PERSIST defines are enabled after termination of the preceding CORRWAIT stage (WAIT) when one of these conditions is true:

- WAIT times out.
- WAIT responds to GO or RESET.
- WAIT ends prematurely because of a PIPEND (non-zero return code).
- DEBUG option occurs for the PERSIST stage. The PERSIST stage activates when the DEBUG option is specified.

If you specify more than one PERSIST stage for a command correlation environment, only the last specified PERSIST stage takes effect.

Messages subject to the DISPLAY action are exposed to user exits, trapping, automation, and logging. For more information about exposure, see “PIPE EXPOSE stage” on page 1-101.

Use the **LIST PERSIST** command to display the status of all enabled PIPE PERSIST elements or use the **STOP PERSIST** command to end an enabled persist.

Messages subject to ROUTE action are routed first, then are exposed as for other messages.

A message subject to COMMAND action is provided as the current message when the indicated command runs. Any output from the command, including the original message, is subject to exposure in the same way as the output of a command issued from the command facility command line. When PERSIST invokes a command, it does so with the same authority as was in effect for the pipeline which established the PERSIST action.

Student exercise

Student exercise



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

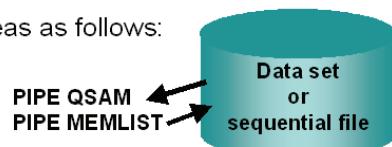
Open your *Student Exercises* book and perform Exercises 1 and 2.

Lesson 3: Using PIPEs to access files

IBM

Lesson 3: Using PIPEs to access files

- **QSAM** stage reads from and writes to areas as follows:
 - Partitioned data sets (PDS)
 - Sequential files
- File name can be one of two forms:
 - Fully qualified data set name, including member name
 - Data definition (DD) name from JCL or ALLOCATE command
- Files do not need to be allocated to NetView
- From-disk (<) and to-disk (>) stages are specialized for NetView files
- You use the **MEMLIST** stage to display list of members in NetView DD or data set



1-39

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **QSAM** stage reads and writes from dynamically allocated data definition (DD) names or data sets. Other devices are also supported when allocated for physical sequential access. The < and > (synonyms for QSAM read and write) can be used only when < and > are immediately followed by a data set name enclosed in quotation marks.

When QSAM is the first stage in a pipeline, the file is read into the pipeline. Otherwise, the contents of the pipeline are written to the file.

The < (from-disk) stage must be the first stage of pipeline. The from-disk stage is limited to a subset of data sets that are allocated to NetView. Issue **BROWSE !** to display the list.



Note: For NLS users, you can use X'5A' to represent the exclamation point character (!).

The **MEMLIST** stage creates a list of members in one or more partitioned data sets (PDS) or data definitions (DD). For a DD, the members are listed for each data set in the concatenation. Members that are defined in an operator data set and members that are defined by using INSTORE COMMON are also listed.

PIPE QSAM stage

IBM

PIPE QSAM stage

```
|-- QSAM ---+-----+-----+-----+  
    +- (DD) ---+  '- data_definition_name-'  
    '- (DSN) -'
```

General purpose read and write utility

- QSAM stage reads from and writes to files:
 - Read when first stage
 - Write otherwise
- Output is series of single-line messages
- Data set name does not need quotes

1-40

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can use the **QSAM** stage with either a data definition (DD) name that is defined by the ALLOCATE command, or a fully qualified data set name. You can enclose a data set name in single quotes. The quotes are ignored.

When specified as a first stage, QSAM reads from the data definition name or data set. When not specified as a first stage, QSAM writes to the data definition name or data set. The messages received on the input stream pass to the output stream.

PIPE < and PIPE > stages

These stages are specialized versions of PIPE QSAM for NetView files. The default data set name is DSIPARM. Additional parameters are supported. INCL reads all included members and resolves any *Data REXX* statements. DISKONLY reads members from disk only. In some cases, files are readable into NetView storage. DISKONLY does not read the in-storage copy.

When you use a data_definition_name of *, NetView searches all standard DD names, if allocated, for the specified member name in the following order:

1. DSICLD
2. DSIPARM
3. DSIPRF
4. DSIVTAM
5. DSIMSG

6. CNMPNL1
7. BNJPNL1
8. BNJPNL2
9. DSILIST
10. DSIOOPEN
11. DSIASRC
12. DSiarpt

Issue **BROWSE !** to display the list of supported standard DD names. For NLS users, you can use X'5A' to represent the exclamation point character (!).

Security considerations

Accomplish security for PIPE QSAM, <, and > stages by implementing security for **READSEC** and **WRITESEC** commands. For example, operator NETOP1 can read from DSIPARM but is not permitted to write to DSIPARM. Implement the stage as follows:

- Issue the READSEC DD=DSIPARM to test read access:

```
DSI633I READSEC COMMAND SUCCESSFULLY COMPLETED
```

- Issue the WRITESEC DD=DSIPARM to test write access:

```
BNH234E 'NETOP1' IS NOT AUTHORIZED TO USE KEYWORD 'DSIPARM'  
BNH235E THE KEYWORD 'DSIPARM' IS PROTECTED BY COMMAND IDENTIFIER  
'*.*.WRITESEC.*' IN 'TBLNAME=SECQSAM'  
DSI213I ACCESS TO 'DSIPARM' IS NOT AUTHORIZED
```

Example Command Authorization Table (CAT) statements for this example are as follows:

```
* Define oper group  
GROUP NVOPS1 NETOP1,NETOP2,OPER1,OPER2,OPER3  
* Restrict access to READSEC  
PROTECT *.*.READSEC.DSIPARM.*  
* Allow NVOPS1 to READSEC  
PERMIT NVOPS1 *.*.READSEC.DSIPARM.*  
  
* Restrict access to WRITESEC  
PROTECT *.*.WRITESEC.*  
PROTECT *.*.WRITESEC.*.*  
* No PERMIT for WRITESEC = no writes allowed
```

PIPE QSAM example

IBM

PIPE QSAM example

Read CNMCMGU (user command definitions)

- PIPE QSAM USER.DSIPARM(CNMCMGU) | COLL | CONS

```
| AOFDA
*** File updated at 08:38:03 on 14 Jun 2006 ***
DDEF.EZLE600A.CMDSYN =  TIMER,TIMERS,TIMR
%INCLUDE MYCMD
```

- Alternatives as follows:
 - PIPE <'USER.DSIPARM(CNMCMGU)' | COLL | CONS
 - PIPE < DSIPARM.CNMCMGU | COLL | CONS
 - PIPE < CNMCMGU | COLL | CONS

1-41

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

In the following example, the pipe reads member CNMCMGU explicitly from the USER.DSIPARM data set. An included member is MYCMD, but the QSAM PIPE displays only the member name.

```
PIPE QSAM USER.DSIPARM(CNMCMGU) | COLL | CONS
```

For comparison, this is a section of a simple REXX EXEC to read members.

```
'TRAP AND SUPPRESS ONLY MESSAGES *'          /* TRAP/SUPPRESS MSGS */
'ALLOCATE DA('DataSetName') SHR FREE'        /* ALLOC/CONNECT FILE */
'WAIT FOR MESSAGES'                         /* WAIT FOR MESSAGES */
'MSGREAD'                                    /* READ A MESSAGE IN */
'TRAP NO MESSAGES'                          /* DISABLE TRAP MSGS */
/* CNM272I ddname IS NOW [ALLOCATED | DEALLOCATED] */
If MSGID() = 'CNM272I' then                 /* allocate worked? */
  DO                                         /* PROCESS CNM272I MSG */
    DDNAME = MSGVAR(1)                      /* SAVE DYNAMIC DDNAME */
    ADDRESS MVS 'EXECIO * DISKR' DDNAME '(STEM Output.'
    Say 'File' DataSetName'/'ddname 'has' output.0 'lines:'
    Do i=1 to Output.0                      /* Output loop */
      SAY SUBSTR(Output.i,1,68)              /* DISPLAY LINE TO USER */
    End                                       /* Output loop */
    Say 'TYPE EXEC is now finished'
  END                                         /* PROCESS CNM272I MSG */
ELSE                                         /* MSG IS CNM272I */
  SAY MSGID() MSGSTR()                      /* DISPLAY MESSAGE */
```

You can use one PIPE statement to replace this REXX program.

Example of PIPE <

IBM

Example of PIPE <

Read CNMCMDU and all included files:
PIPE < CNMCMDU INCL DISKONLY | COLL | CONS

```
| AOFDA
*** File updated at 08:38:03 on 14 Jun 2006 ***
DDEF.EZLE600A.CMDSYN = TIMER,TIMERS,TIMR
* Member MYCMD
CMDDEF.MINE.MOD=DSICCP
CMDDEF.MINE.ECHO=N
CMDDEF.MINE.TYPE=R
```

1-42

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

In the example, the pipe reads CNMCMDU, using the from-disk stage. The standard NetView DD names are searched and the member found in the DSIPARM library concatenation. The included member (MYCMD) is also read into the pipeline because the INCL option is specified on the from-disk stage.

PIPE < CNMCMDU INCL DISKONLY | COLL | CONS

MEMLIST stage example

IBM

MEMLIST stage example

- MEMLIST lists members in NetView DD name or data set
Output: Multiline message, one line per member
- Example: Display all NetView command lists and EXECs
PIPE MEMLIST DSICLD | CONS

```
CNME1505 0
CNME1096 0
FKXEDVPT 0
FKXEDVPA 0
...
SETCG    1
TSTCONN  1
...
CNME1048 6
CNME1049 6
```

1-43

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **MEMLIST** stage lists members in NetView DD name or data set. The output is a multiline message with one line for each member as follows:

- Columns 1 to 8: Member name
- Column 10: Relative data set number
 - -1 if file is found in an operator data set
 - 0 if file is loaded in storage
 - 1 when a data set is specified

For DD name, number matches concatenation order shown by LISTA command

MEMLIST includes operator data sets and members that are loaded in storage by INSTORE and MEMSTORE.

Student exercise

IBM

Student exercise



1-44

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 3.

Lesson 4: Pipelines and variables



Lesson 4: Pipelines and variables

- Pipeline output can be stored in variables
- When specified as the first stage, variable data can be read into the pipeline
- Stages for variables are as listed. Each can process local, task global, or common global variables
 - VAR
 - STEM
 - VARLOAD

1-45

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

With the **VAR** stage, you can read records from or write records to variables in a command procedure variable pool. The **\$VAR** stage is similar to VAR. However, \$VAR also reads or writes the VIEW attribute variables (which start with \$) that are associated with the specified data variables.

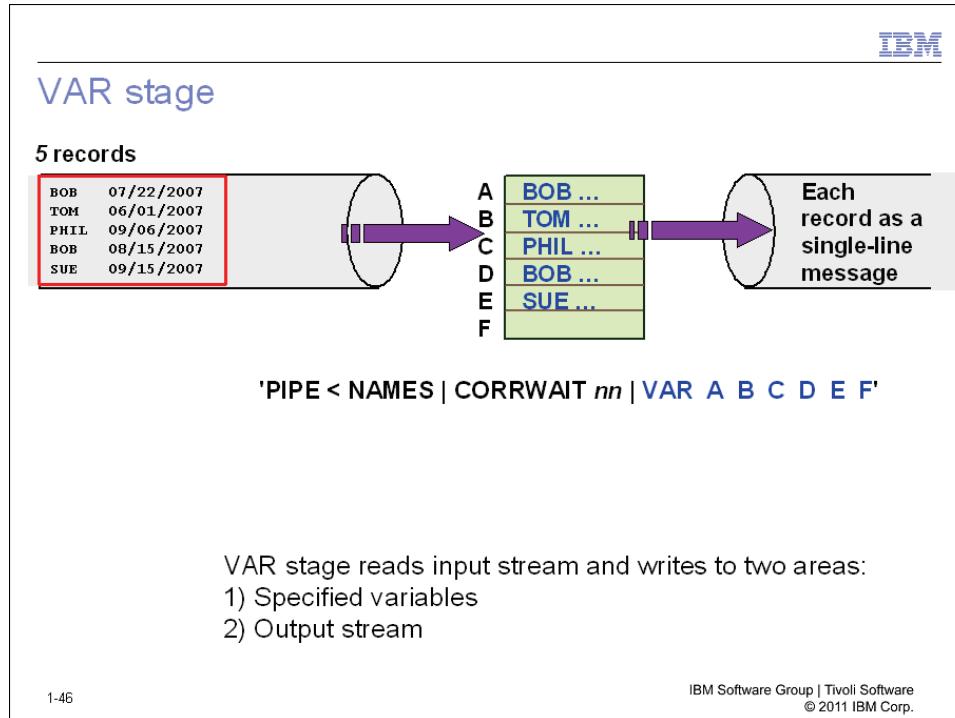
With the **STEM** stage, you can read or write an array of stemmed command procedure variables. The **\$STEM** stage is similar to STEM. However, \$STEM also reads or writes the VIEW attribute variables (which start with \$) that are associated with the specified array of stemmed data variables.

With the **VARLOAD** stage, you can set values for variables that pass in the input stream. The names and values of the variables that VARLOAD sets are specified by the records that passed on the primary input stream. VARLOAD sets one variable for each input message that contains a character other than a blank or asterisk in the first position of the record.



Note: You cannot issue these stages from the NetView command line.

VAR stage



The **VAR** stage reads data from the input stream, sets the specified variables, and writes the data to the output stream. In this example, the NAMES file is read into the pipeline. The VAR stage is passed five records, setting five variables A, B, C, D, and E. Variable F is not set because only five records were passed. The VAR stage syntax is as follows:

```
-----.
      .- ( 0 )-----.   V          |
| --+--VAR---+-----+-----name-----+-----|
  '-$VAR-'    +- ( COMMON ) -+
           +- ( TASK ) ---+
           '- ( number ) -'
```

When VAR is the first stage of a pipeline, records are read from the variable that is specified. Each record passes as a single-line message to the pipeline output stream.

When VAR is specified as a subsequent stage, messages are read from its input stream and written to both the specified variables and to its output stream. Data from the first input message is placed

in the first variable and written to the output stream. Data from the second message goes in the second variable and to the output stream. The data flows similarly for the rest of the process.

- (COMMON): Specifies to access the common global variable dictionary instead of the local dictionary for the procedure.
- (TASK): Specifies to access the task global variable dictionary instead of the local variable dictionary for the procedure.
- (*number*): Specifies the number of invocations (generations) to refer back when setting the variables. The default is zero (0).
- *name* - Specifies the name of the variable to read from or write to.



Tip: Pipeline variables are dropped before being used. This can cause errors if you use SIGNAL ON NOVALUE conditions in REXX programs. Use REXX SYMBOL() function to test for null variables: An example follows:

```
IF Symbol(E) <> 'LIT' then /* If variable E is not null */
```

PIPE VAR example

PIPE VAR example



Parse terminal name from **LIST** " command response:

First line: STATION: NETOP1 TERM: A01A701

```
/* REXX */

'PIPE NETV LIST '' ',      /* Issue LIST command */
' | TAKE FIRST    ',      /* take only line 1   */
' | VAR LINE1      ',      /* save in local var */

PARSE VAR LINE1 . 'TERM:' Oper_terminal

Say 'Your terminal is:' Oper_terminal
```

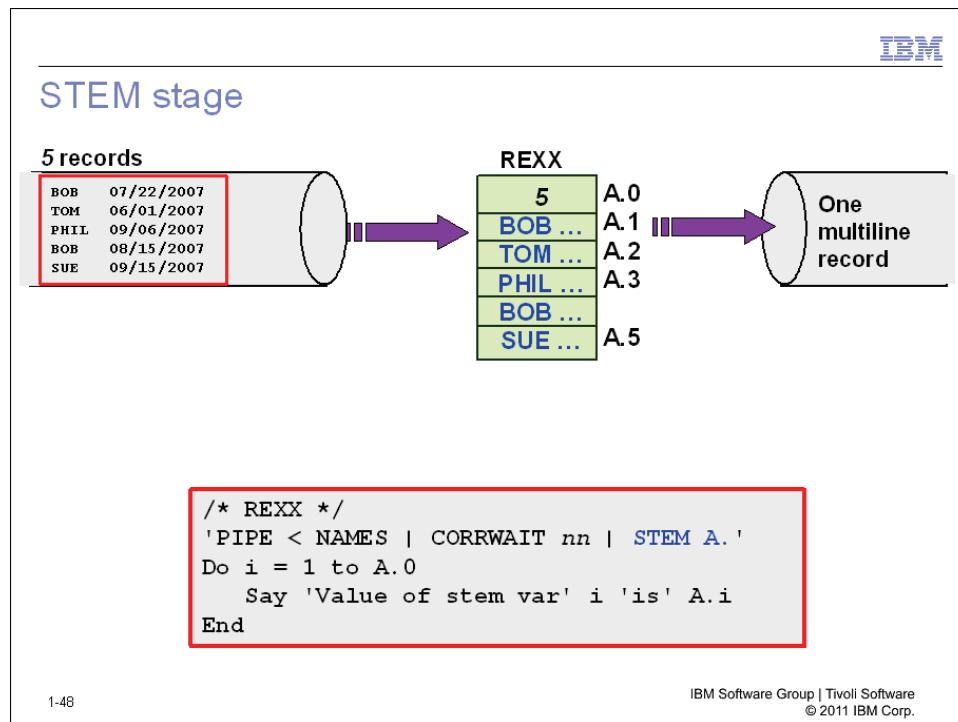
Example output: Your terminal is: A01A701

1-47

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

In this example, the NetView **LIST** " command is issued. Several single-line messages are generated. The pipeline takes the first message and places it into the LINE1 variable without stopping at the other messages. The next instruction, REXX **PARSE VAR**, parses LINE1 for the operator terminal.

STEM stage



The **STEM** stage reads data from the input stream, sets the specified stem variables, and writes the data to the output stream. In this example, the **STEM** output would be as follows:

```
Value of stem var 1 is BOB 07/22/2007
Value of stem var 2 is TOM 06/01/2007
Value of stem var 3 is PHIL 09/06/2007
Value of stem var 4 is BOB 08/15/2007
Value of stem var 5 is SUE 09/15/2007
```

- When STEM is the first stage, it reads records from an array of stemmed variables. Each record passes as a single-line message to the pipeline output stream.
- When STEM is used for reading records into a stem variable, the count of stem variables automatically updates in element zero.
- When STEM is not the first stage, it writes each line of each a message to stem variables and to the output stream. In addition, an integer appends to the given variable name (for example, VARNAME n).
- When STEM is used for writing records to the pipeline, the number of records to write must be defined in element zero.

The STEM stage syntax is as follows:

```
. - (0)-----.
>>+-----+-----+-----+-----+-----+-----+
    +-STEM--+- (number) --+-----+-----+-----+
    '-$STEM-'  +- (COMMON) --+
                  ' - (TASK) ---'

. - FROM 1-----.
>-+-----+-----+-----+-----+-----+-----+><
    +- FROM frnumber+-+-----+-----+-----+
    ' - APPEND-----'
```

- *stemroot*: Specifies the name of the stem variable to read from or write to. It should end with a period (.) if you use a REXX EXEC.
- APPEND: Specifies that new data should be appended as additional stem variables at the end of the current stem variables. APPEND can be used only on a stage that is not first.
- COLLECT: Causes STEM to build one multiline message instead of many single-line messages. COLLECT is allowed only when STEM is the first stage in a pipeline.
- FROM: Indicates a starting point for access to the stem variables.

Student exercise

Student exercise



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 4.

Lesson 5: Working with text in pipeline



Lesson 5: Working with text in pipeline

- COLOR
- LOCATE and NLOCATE
- TOSTRING
- TAKE and DROP
- COLLECT and SEPARATE
- CHOP
- BETWEEN
- REVERSE
- PICK
- SUBSYM
- CHANGE
- CASEI
- SPLIT
- STRIP
- SORT
- JOINCONT
- DUPLICAT
- DELDUPES
- COUNT

1-50

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This lesson contains information on several pipeline stages for modifying the contents of pipeline data. Use the NetView HELP PIPE STAGES command to display a panel with a list of all NetView pipe stages. On the panel, **tab over** to a stage name (for example, LOCATE) and press Enter to view the help for the LOCATE stage.

Manipulated pipeline data routes to the primary output stream of these pipe stages with all other data sent to the secondary output stream.

COLOR Stage: Change message attributes

IBM

COLOR stage: Changing message attributes

PIPE NETV LIST DSILog | COLOR BLUE REV | CONSOLE

```
NetView V6R1M0          Tivoli NetView   AOFDA NETOP1  06/13/11 19:47:27
* AOFDA    PIPE NETV LIST DSILog | COLOR BLUE REV | CONSOLE
- AOFDA    TYPE: OPT TASKID: DSILog  TASKNAME: DSILog  STATUS: ACTIVE
- AOFDA    MEMBER: DSILogK
- AOFDA    PRIMARY:DSILOG STATUS:ACTIVE  SECONDARY:DSILOGS STATUS:INACTIVE
- AOFDA    AUTOFLIP: YES             RESUME: YES
- AOFDA    LOADMOD: DSIZDST
- AOFDA    Task Serial: 332 REXX Environments: 1 (1%)
- AOFDA    Messages Pending: 0 Held: 0
- AOFDA    WLM Service Class: Not Available
- AOFDA    END OF STATUS DISPLAY
```

Modify the entire multiline response to display as reverse video blue

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **COLOR** (presentation attributes) stage changes how messages look at the NetView console. You can change the color, highlighting, and intensity of data records in a pipeline:

- Color: Blue, red, pink, yellow, green, white, turquoise, default
- Highlighting: Reverse, underscore, blink, none
- Intensity: Normal, bright, dark

Synonyms of COLOR are COLOUR and PRESATTR.

LOCATE stage: Selecting data from pipeline

IBM

LOCATE stage: Selecting data from pipeline

Pipe NETV List Status=Tasks | LOCATE 55.10 /NOT ACTIVE/ | CONSOLE CLEAR

NetView V6R1MO	Tivoli NetView	AOFDA NETOP1	06/13/11 19:50:48
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSITRACE	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSIELTSK	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSICORSV	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: ALIASAPL	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSIAL2WS	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSIAOPT	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSIB2MT	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSIGBS	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSIKREM	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSIROWS	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSIRTR	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSIMPTSK	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: VPDTASK	STATUS: NOT ACTIVE

Locate only NetView tasks that are NOT ACTIVE (position 55, length 10)

1-52

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **LOCATE** stage selects messages that match a specified delimited character string to be passed to the primary output stream. Messages that do not contain the character string pass to the secondary output stream, if connected. If no secondary output stream exists, the messages that do not match the LOCATE are discarded.

LOCATE examines each input message for a match to the delimited string. You can supply a position and length pair to limit the search to a particular column range. An example follows:

```
'Pipe NETV List Status=Tasks',
' | LOCATE 55.10 /NOT ACTIVE/ | CONSOLE CLEAR'
```

This stage issues the NetView **L**I**S**T **S**TATU**S=T**ASKS command and places the output in the pipeline. Only records that contain the string NOT ACTIVE beginning in position 55 for a length of 10 are selected.

NLOCATE stage: Discarding data from pipeline

IBM

NLOCATE stage: Discarding data from pipeline

Pipe NETV List Status=Tasks | **NLOC 55.10 /NOT ACTIVE/ /OPT/ /PPT/ /MNT/ | LOC /AUTO/ | COLOR GRE | CONSOLE CLEAR**

NetView V6R1MO	Tivoli NetView	AOFDA NETOPI	06/13/11 19:54:55
- AOFDA	TYPE: OST TASKID: AUTOAQN	RESOURCE: AUTOAQN	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: AUTOC75	RESOURCE: AUTOC75	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: AUTOC76	RESOURCE: AUTOC76	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: AUTOC77	RESOURCE: AUTOC77	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: AUTODC1	RESOURCE: AUTODC1	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: AUTODC2	RESOURCE: AUTODC2	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: AUTODC3	RESOURCE: AUTODC3	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: AUTODC4	RESOURCE: AUTODC4	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: AUTOEDAT	RESOURCE: AUTOEDAT	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: AUTONALC	RESOURCE: AUTONALC	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: AUTONVSP	RESOURCE: AUTONVSP	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: AUTOPSAV	RESOURCE: AUTOPSAV	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: AUTOTMSI	RESOURCE: AUTOTMSI	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: AUTOXCF	RESOURCE: AUTOXCF	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: AUTOXDSC	RESOURCE: AUTOXDSC	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: AUTO1	RESOURCE: AUTO1	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: DBAUTO1	RESOURCE: DBAUTO1	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: DBAUTO2	RESOURCE: DBAUTO2	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: AUTO2	RESOURCE: AUTO2	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: AUTOTCP_C	RESOURCE: AUTOTCP_C	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: AUTOPRTS	RESOURCE: AUTOPRTS	STATUS: ACTIVE
- AOFDA	TYPE: OST TASKID: AUTOTCP_S	RESOURCE: AUTOTCP_S	STATUS: ACTIVE

Locate subset of NetView autotasks that are ACTIVE

1-53

IBM Software Group | Tivoli Software
 © 2011 IBM Corp.

The **NLOCATE** stage discards messages from the primary output stream that match a specified delimited character string. Messages that do not contain the character string are passed to the primary output stream. Messages that contain the character string are passed to the secondary output stream, if connected.

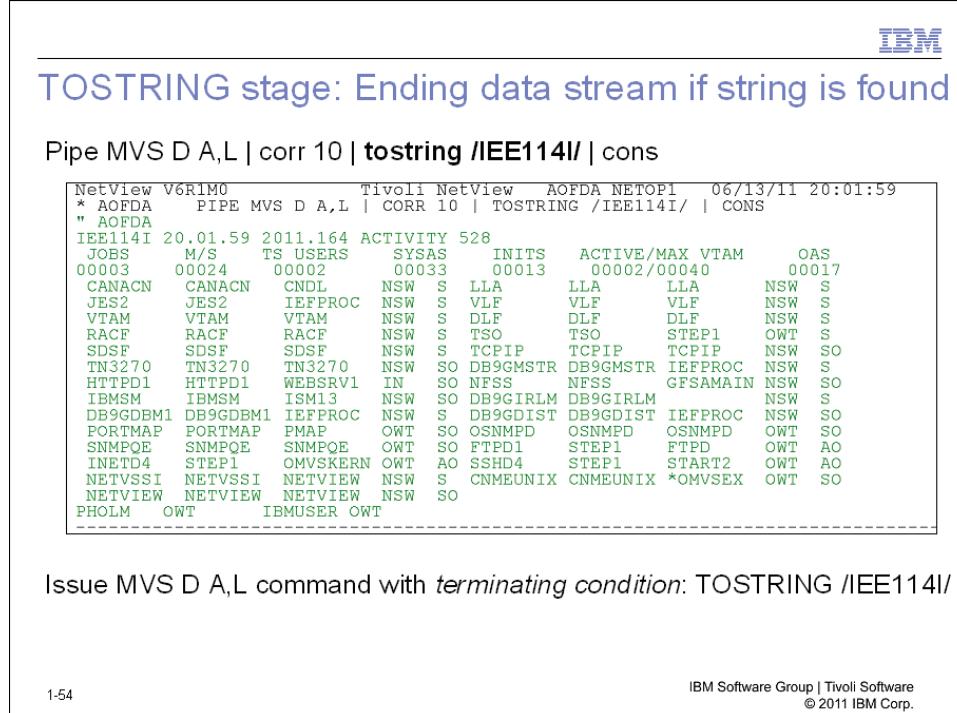
You can supply a position and length to limit the search to a particular column range. An example follows:

```
'Pipe NETV List Status=Tasks',
    '|NLOC 55.10 /NOT ACTIVE/ | COLOR GRE |CONSOLE CLEAR'
```

This stage issues the NetView **LIST STATUS=TASKS** command and places the output in the pipeline. Only the records that do not contain the string NOT ACTIVE beginning in position 55 for a length of 10 are selected.

The example on the slide contains other data to NLOCATE. In this case, data for all tasks of type OPT, PPT, and MNT are to be discarded. The LOCATE stage further refines the pipeline data by finding tasks that begin with AUTO.

TOSTRING stage: Ending data stream if string is found



With the **TOSTRING** stage, you can select messages containing the text that match a specified string. Selected messages pass to the primary output stream. Those not selected pass to the secondary output stream, if connected. TOSTRING generates a *terminating condition* for the CORRWAIT stage. TOSTRING ends the preceding CORRWAIT stage.

In the example, if you do not specify a terminating condition, the IEE114I is displayed, but the pipeline waits the full 10 seconds before it disconnects and moves to the CONS stage.

You can specify up to 40 delimited strings, each with an optional position and length pair to limit the column range of the search.

TAKE stage: Selecting data based on line number

IBM

TAKE stage: Selecting data based on line number

PIPE NETV PING 10.44.15.200 | CORR 10 | **TAKE LAST 1** | CONS

C AOFDA BNH765I Pinging tived1.tived.ibm.com at 10.44.15.200 with 3 packets
of length 16 bytes
C AOFDA BNH767I 16 bytes received from 10.44.15.200: seq=1 in 0ms
C AOFDA BNH767I 16 bytes received from 10.44.15.200: seq=2 in 1ms
C AOFDA BNH767I 16 bytes received from 10.44.15.200: seq=3 in 0ms
C AOFDA BNH769I 3 packets sent, 3 packets received, 0.00% packet loss
C AOFDA BNH770I Round trip times from 0 to 1 ms, averaging 0ms

Issue NetView **PING** command. Keep only the last message, BNH770I, in output stream. All other messages are to be discarded (sent to secondary output stream)

1-55

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

With the **TAKE** stage, you can specify the number of messages or lines that pass to the primary output stream (if any). All messages or lines that exceed this number pass to the secondary output stream.

TAKE disconnects the input stream with discarded messages that pass to the secondary output stream. TAKE generates a *terminating condition* for the CORRWAIT stage. TAKE LAST *nn* waits until the last message is displayed (or a timeout occurs).

DROP stage: Discarding data based on line number

IBM

DROP stage: Discarding data based on line number

Pipe netv DISPPPI ALL | corr | sep | **drop first 3 | drop last 1** | coll | cons clear

NetView V6R1MO	Tivoli NetView	AOFDA NETOP1	06/13/11 20:06:02
* AOFDA			
DW0951I NETVALRT	ACTIVE	1000	0 0043
DW0951I DSICQSK	ACTIVE	100	0 0043
DW0951I AOFDAHIM	ACTIVE	1000	0 0043
DW0952I NETVRCV	INACTIVE	2	0
DW0951I CNMUNIX	ACTIVE	1000	0 0042
DW0951I CNMPCMDR	ACTIVE	2000	0 0043

Issue NetView **DISPPI ALL** command. Discard the first three lines (header) and last line. All other messages are to be kept and collected into a single multiline message

1-56 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

With the **DROP** stage, you can specify how many messages or lines are to be discarded from the primary output stream. When the specified number of messages are read from the input stream and discarded, all other messages copy to the primary output stream. Discarded messages or lines pass to the secondary output stream, if connected.

COLLECT and SEPARATE stages: Processing messages

IBM

COLLECT and SEPARATE stages: Processing messages

Pipe netv DISPPI ALL | corr | **sep** | drop first 3 | drop last 1 | **collect** | cons

```
NetView V6R1MD      Tivoli NetView   AOFDA NETOP1  06/13/11 20:10:27
* AOFDA  PIPE NETV DISPPI ALL | CORR | SEP | DROP FIRST 3 | DROP LAST 1 |
    COLLECT | CONS

' AOFDA
DWO951I NETVALRT ACTIVE      1000      0      0      0 0043
DWO951I DSQITSK  ACTIVE      100       0      0      0 0043
DWO951I AOFDAHTM ACTIVE     1000      0      0      0 0043
DWO952I NETVRCV  INACTIVE     2        0      0      0 0043
DWO951I CNMEEUNIX  ACTIVE    1000      0      3      0 0042
DWO951I CNMPCMDR ACTIVE    2000      0      0      0 0043
```

Issue NetView **DISPPI ALL** command. **SEPARATE** multiline response.
Discard first three lines (header messages) and last line. All other messages (DWO951I) are to be kept as a single multiline message with a **COLLECT** stage

1-57

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **SEPARATE** stage transforms multiline messages into multiple single-line messages. Input single-line messages pass without modification. Output single-line messages inherit all of the attributes of the input messages that created them. The output of **SEPARATE** consists of single-line messages.

The **COLLECT** stage creates a multiline message from one or more messages in the pipeline. The attributes of the output are inherited from the first line that is collected for that message.

CHOP stage: Truncating pipeline data

IBM

CHOP stage: Truncating pipeline data

Pipe NETV List Status=Tasks | LOCATE 55.10 /ACTIVE/ | LOCATE 19.3 /AUT/ | **CHOP 46** | CONSOLE CLEAR

```
TYPE: OST TASKID: AUTDVIPA RESOURCE: AUTDVIPA
TYPE: OST TASKID: AUTOAON RESOURCE: AUTOAON
TYPE: OST TASKID: AUTODC1 RESOURCE: AUTODC1
TYPE: OST TASKID: AUTODC2 RESOURCE: AUTODC2
TYPE: OST TASKID: AUTODC3 RESOURCE: AUTODC3
TYPE: OST TASKID: AUTODC4 RESOURCE: AUTODC4
TYPE: OST TASKID: AUTODC5 RESOURCE: AUTODC5
...
TYPE: OST TASKID: AUTTCP9 RESOURCE: AUTTCP9
TYPE: OST TASKID: AUTTCP10 RESOURCE: AUTTCP10
TYPE: OST TASKID: AUTTRAP RESOURCE: AUTTRAP
TYPE: OST TASKID: AUTWKSTA RESOURCE: AUTWKSTA
```

Issue **LST STATUS=TASKS** command. Locate a subset of NetView autotasks that are ACTIVE. Delete **STATUS: ACTIVE** text from each line

1-58

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **CHOP** stage truncates lines after a specified column, character, or string. The data that CHOP keeps passes to the primary output stream. The data that CHOP discards passes to the secondary output stream, if connected. In this example, all data after column 46 is deleted. This action removes STATUS: ACTIVE from each line in the pipeline.

BETWEEN stage: Dividing a message stream

IBM

BETWEEN stage: Dividing a message stream

```
Pipe < CNMPNL1.EUYSLIST | TAKE 90
| BETWEEN 1.14 :IF DTYP=MSG/ 1.6 :ENDIF/ | CONS ONLY
```

```
CNWKWIND OUTPUT FROM PIPE < CNMPNL1.EUYSLIST | TAKE 90 | LINE 0 OF 24
----- Top of Data -----
: IF DTYP=MSG/
  Enter HELP PIPE DSIVSAM to display syntax and general usage.
  Enter HELP DSIVSAM DEL to receive help on deleting records.
  Enter HELP DSIVSAM GET to receive help on reading VSAM records.
  Enter HELP DSIVSAM GETREV to receive help on reading VSAM records in
    reverse sequence.
  Enter HELP DSIVSAM INQUIRE to receive help on displaying data set
    characteristics for a file.
  Enter HELP DSIVSAM PUT to receive help on creating or replacing records.
:ENDIF
: IF DTYP=MSG/
  Enter HELP PIPE DSIVSMX to display syntax and general usage notes.
  Enter HELP DSIVSMX CLOSE to receive help on closing records.
  Enter HELP DSIVSMX DEL to receive help on deleting records.
  Enter HELP DSIVSMX GET to receive help on reading records.
  Enter HELP DSIVSMX GETREV to receive help on reading records in
    reverse sequence.
  Enter HELP DSIVSMX IDCAMS to receive help on data set maintenance.
  Enter HELP DSIVSMX INQUIRE to receive help on displaying data set
    characteristics for a file.
  Enter HELP DSIVSMX OPEN to receive help on opening files locally to the
    issuing task.
  Enter HELP DSIVSMX PUT to receive help on putting records.
:ENDIF
----- Bottom of Data -----
```

Read member EUYSLIST. Keep all data between :IF DTYP=MSG/ and :ENDIF statements

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **BETWEEN** stage divides a message stream into sections. The selected sections begin with a message containing a specified string and end with either a specified string or a number of messages. The selected sections pass to the primary output stream. The sections that were not selected pass to the secondary output stream. All lines in EUYSLIST between the :IF DTYP=MSG/ and :ENDIF statements are included in the pipeline data.

REVERSE stage: Reversing order of pipeline data

IBM

REVERSE stage: Reversing order of pipeline data

Pipe MVS D A,L | CORR | take first 1 | reverse message | cons clear

```
" AOFDA
ITEMUSER OWT
CANSNA CANSNA AGENT NSW SO
OSNMPD OSNMPD OWT SO CANSDSST CANSDSST TEMS NSW SO
SYSLOGD8 STEP1 OMVSKERN NSW AO RXSERVE RXSERVE RXSERVE OWT SO
TCP IP TCP IP TCPIP NSW SO INETD4 STEP1 OMVSKERN OWT AO
RACF RACF RACF NSW S TSO TSO STEP OWT S
AUTOSSI AUTOSSI NETVIEW NSW S AUTONETV AUTONETV NETVIEW NSW SO
RMF RMF IEFPROC NSW S RRS RRS RRS NSW S
JES2 JES2 IEFPROC NSW S VTAM VTAM VTAM NSW S
LLA LLA LLA NSW S VLF VLF VLF NSW S
00002 00015 00001 00028 00007 00001/00300 00009
JOBS M/S TS USERS SYSAS INITS ACTIVE/MAX VTAM OAS
IEE114I 15.19.30 2007.249 ACTIVITY 425
```

Issue MVS D A,L command. Reverse output order. No need to separate message

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can use **REVERSE** stage to reverse the order of data in the pipeline as follows:

- LINE (default): Specifies that each line of output is to be reversed, character by character. For example, DSI069I SYNTAX ERROR changes into RORRE XATNYS I960ISD.
- MESSAGE: Specifies that each multiline message is to be reversed, line by line. Affects only multiline messages.
- STREAM: Specifies that the next stage receives messages in reverse order. This option has no effect on the structure of multiline messages.
- REVERSE waits until all messages are in the pipeline before processing the data.

PICK stage: Selecting specific data

IBM

PICK stage: Selecting specific data

```
PIPE NETV TASKUTIL | SEP | DROP FIRST 3 | DROP LAST 5
| PICK 20.7 > / 11 | COLL | CONS
```

		Tivoli NetView	AOFDA	NETOPI	06/13/11 20:18:53
* AOFDA	/	PIPE NETV TASKUTIL SEP DROP FIRST 3 DROP LAST 5 PICK 20.7 >			
		1/ COLL CONS			
AOFDA					
AOFDAPPT	PPT 255	84.51	0.00	0.00	0 577 **NONE**
AUTOAON	AUTO 250	44.68	0.00	0.00	0 548 **NONE**
AUTO1	AUTO 250	13.54	0.00	0.00	0 332 **NONE**
AUTO2	AUTO 250	13.62	0.00	0.00	0 272 **NONE**

CPU time, in seconds

Issue **TASKUTIL** command and select only the lines for tasks that have greater than 1 second of CPU time (column 20, length of 7)

1-61

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **PICK** stage selects messages satisfying a criteria and passes them to the primary output stream. Messages that do not meet the specified criteria pass to the secondary output stream, if connected.

The **PICK** stage examines only the first line of a multiline message. If selected, the entire multiline message passes to the primary output stream. Use the **SEParate** stage to break a multiline message into multiple single-line messages.

Specify the selection criteria by giving a *position.length* within the message line to be compared against: another *position.length* within the message line or a character string. In this example, the **TASKUTIL** output is interrogated for any task that has consumed more than 1 second of CPU time. Because the field being checked has a length of seven, the string that is specified on **PICK** must contain seven characters between the delimiters. The delimiters are forward slashes.

SUBSYM stage: Substituting symbolic variables



SUBSYM stage: substituting symbolic variables

```
NETVASIS Pipe LIT /System name =  
  &SYSNAME/  
  | LIT /NetView Domain =  
    &DOMAIN/  
  | LIT /TCP name = &CNMTCPN/  
  | LIT /Network ID = &CNMNETID/  
  | SUBSYM | CONS CLEAR
```

```
NetView V6R1MO          Tivoli NetView   AOFDA NETOP1  06/14/11 13:58:39  
| AOFDA     Network ID=ADCD  
| AOFDA     TCP name = TCPIP  
| AOFDA     NetView Domain = AOFDA  
| AOFDA     System name = ADCD1
```

Insert four literal strings in the pipeline. Use **SUBSYM** to display value of &sysname, &domain, &cnmnetid, and &cnmtcpn symbolic variables

1-62

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **SUBSYM** stage substitutes system or user-defined symbolic variables (including NetView &DOMAIN) into messages in the pipeline. In this case, NETVASIS is used for retaining the mixed case of the literal strings also. To preserve the order of the strings, you can insert a REVERSE STREAM before the CONSOLE stage.

CHANGE stage: Replacing string data in pipeline

IBM

CHANGE stage: Replacing string data in pipeline

Pipe netv LIST " | change /NETOP1/ /NONAME/ | cons

```
NetView V6R1MO      Tivoli NetView   AOFDA NETOP1  06/14/11 14:06:34
* AOFDA PIPE NETV LIST '' | CHANGE /NETOP1/ /NONAME/ | CONS
- AOFDA STATION: NONAME TERM: SCOTCP19
- AOFDA HCOPY: NOT ACTIVE PROFILE: DS1PROFB
- AOFDA STATUS: ACTIVE IDLE MINUTES: 0
- AOFDA ATTENDED: YES CURRENT COMMAND: PIPE
- AOFDA AUTHRCVR: YES CONTROL: GLOBAL
- AOFDA NGMFADMN: YES DEFAULT MVS CONSOLE NAME: NET1POAD
- AOFDA NGMFVSPN: NNNNN (NO SPAN CHECKING ON NMC VIEWS)
- AOFDA NGMFCMDS: YES AUTOTASK: NO
- AOFDA IP ADDRESS: N/A
- AOFDA OP CLASS LIST: NONE
- AOFDA DOMAIN LIST: AOFDA (I) AOFDB (I)
- AOFDA ACTIVE SPAN LIST: NONE
- AOFDA Task Serial: 14649 REXX Environments: 2 (1%)
- AOFDA Messages Pending: 0 Held: 0
- AOFDA WLM Service Class: Not Available
- AOFDA END OF STATUS DISPLAY
```

Issue **LIST "** command and change the operator name from NETOP1 to NONAME

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **CHANGE** stage replaces occurrences of the first specified string with the second string. Either string can be null. If the first string is null, the second string inserts at the beginning of each line. If the second string is null, all occurrences of the first one are deleted. Data between substitutions are copied without change.

If a secondary output stream is defined, changed messages pass to the primary output stream. Unchanged messages (because first string is not found) pass to the secondary output stream.

In this example, operator NETOP1 issues the **LIST "** command. Notice the **STATION: NONAME** in the first line of the response.

CASEI stage: comparing strings

IBM

CASEI stage: Comparing strings

```
NETVASIC PIPE LITERAL /ABcdefghi/
| LITERAL /abc/
| LITERAL /xyz/
| CASEI LOCATE /abC/
| CONSOLE
```

```
NetView V6R1M0          Tivoli NetView    AOFDA NETOP1   06/14/11 14:13:00
* AOFDA      PIPE LITERAL /ABcdefghi/ | LITERAL /abc/ | LITERAL /xyz/ | CASEI
|           LOCATE /abC/ | CONS
| AOFDA     abc
| AOFDA     ABcdefghi
```

Display strings that contain the characters ABC regardless of case

1-64

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **CASEI** stage compares character strings without case sensitivity to uppercase or lowercase EBCDIC characters. In this example, CASEI LOCATE /abC/ matches *ABcdefghi* and *abc*.

SPLIT stage: Dividing data based on string

IBM

SPLIT stage: Dividing data based on string

```
PIPE LITERAL /BUY IT, YOU'LL SPLIT AND LIKE IT BETTER/
| SPLIT 3 AFTER STRING /IT/
| CONSOLE

NetView V6R1M0          Tivoli NetView   AOFDA NETOP1  06/14/11 14:20:18
* AOFDA    PIPE LITERAL /BUY IT, YOU'LL SPLIT AND LIKE IT BETTER/ | SPLIT 3
*          AFTER STRING /IT/ | CONSOLE
| AOFDA    BUY IT, Y
| AOFDA    + OU'LL SPLIT AN
| AOFDA    + D LIKE IT BE
| AOFDA    + TTER
```

Divide the text string, BUY IT, YOU'LL SPLIT AND LIKE IT BETTER,
three characters after each occurrence of the string IT

1-65

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **SPLIT** stage divides a line of text into multiple lines, based on a specified string. The default delimiter is a blank. The **SPLIT 3 AFTER STRING /IT/** example displays the following text:

```
BUY IT, Y
OU'LL SPLIT AN
D LIKE IT BE
TTER
```

STRIP stage: Removing characters from a line

IBM

STRIP stage: Removing characters from a line

Pipe < LOGTBL | **strip trailing not // 8** | coll | cons

```
* LOGTBL: Lab Auto Tbl
* DSI547I DSILOG : SECONDARY VSAM DATA SET IS NOW ACTIVE
* DSI546I DSILOG : PRIMARY VSAM DATA SET IS NOW ACTIVE
*
*
if msgid='DSI546I' | msgid='DSI547I' & token(2) = 'DSILOG'
& token(2) = 'DSILOG' & token(4) = PriOrSec then
  EXEC(CMD('LOGAUTO ' PriOrSec) ROUTE(ONE *)) HOLD(Y);
```

Use **PIPE <** to read an automation table (for example, LOGTBL). Use **STRIP** to remove the sequence numbers in columns 73 to 80

1-66

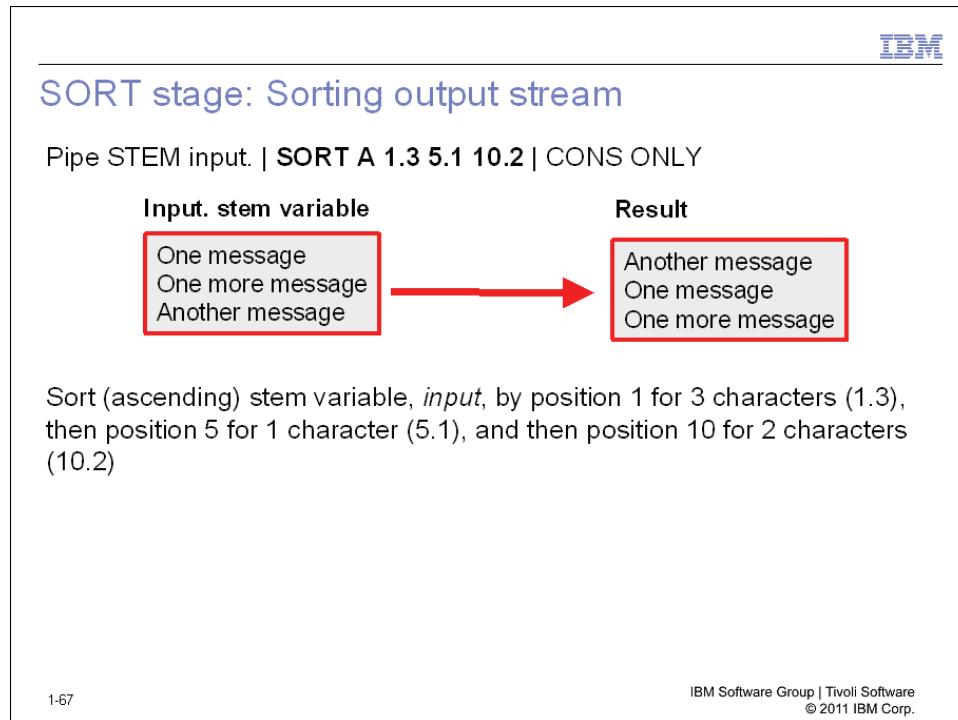
IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **STRIP** stage removes blanks or other specified characters from the beginning or end of message data. Alternately, STRIP removes all characters up to a blank or other specified characters.



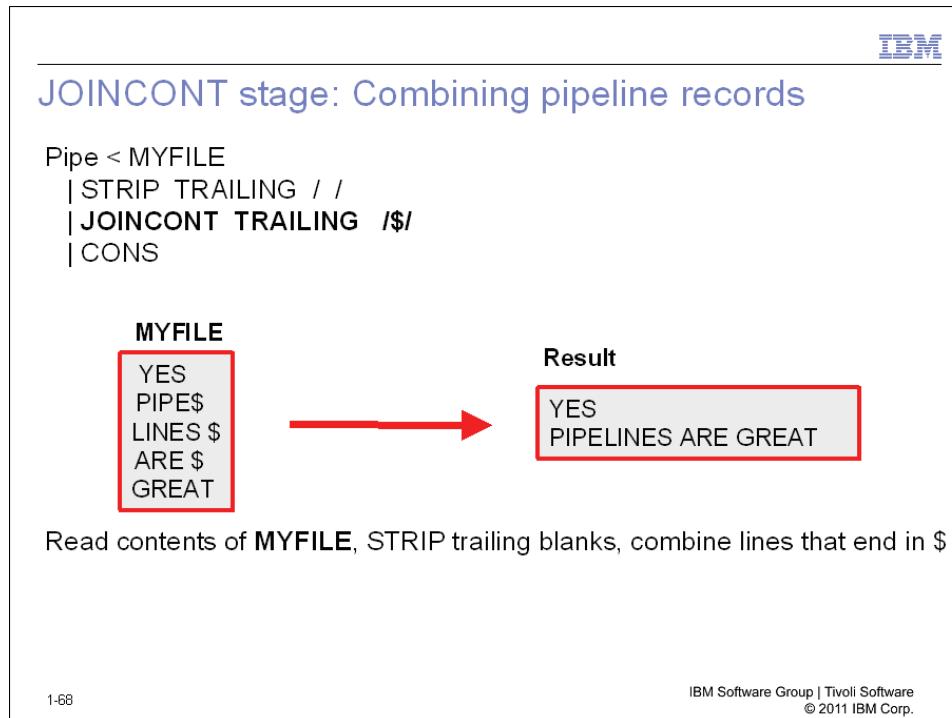
Tip: STRIP can remove unwanted blanks or other characters before you use the JOINCONT stage. Discussion about JOINCONT occurs later.

SORT stage: Sorting output stream



The **SORT** stage reads messages from the input stream and writes them to the output stream in a specified order, ascending or descending. Only the first line of each multiline message is examined. To sort lines within a multiline message, the **SEPARATE** stage must be included before **SORT**. If messages contain identical sort fields, they retain their input stream order when passed to the output stream.

JOINCONT stage: Combining pipeline records



The **JOINCONT** stage joins consecutive messages in the pipeline when a match to a specified string is found. A message is considered in its entirety, and it can include blanks or even sequence numbers if reading from a file. In this example, the pipeline records that contain a \$ are joined. **MYFILE** contains five lines. The result contains two lines.

DUPLICAT stage: Duplicating pipeline records

IBM

DUPLICAT stage: Duplicating pipeline records

Pipe LIT /MSG NETOP1,HELLO/ | DUP 3 | CORRCMD /AUTO1:
| COLOR BLUE REV | CONS

```
tView V6R1MO          Tivoli NetView   AOFDA NETOP1  06/14/11 14:33:12
AOFDA    PIPE LIT /MSG NETOP1,HELLO/ | DUP 3 | CORRCMD /AUTO1: | COLOR BLUE
         REV | CONS
AOFDA    DS1039I MSG FROM AUTO1 : HELLO
AOFDA    DS1001I MESSAGE SENT TO NETOP1
AOFDA    DS1039I MSG FROM AUTO1 : HELLO
AOFDA    DS1001I MESSAGE SENT TO NETOP1
AOFDA    DS1039I MSG FROM AUTO1 : HELLO
AOFDA    DS1001I MESSAGE SENT TO NETOP1
AOFDA    DS1039I MSG FROM AUTO1 : HELLO
AOFDA    DS1001I MESSAGE SENT TO NETOP1
```

Run the **MSG** command four times on task AUTO1. Output returns to originating operator

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **DUPLICAT** stage takes the messages in the input stream, copies them, and writes the copied messages to the output stream. The copies are marked as *copy* rather than *primary*. The message text does not change. In this example, the output from the pipeline is the DS1001I message. The DS1039I message results from the MSG NETOP1,HELLO command.

DELDUPES stage: Removing duplicate records

IBM

DELDUPES stage: Removing duplicate records

Pipe < LOGTIMES | SORT 1.18
| DELDUPES KEEPLAST 1.18 | CONS

LOGTIMES

DOE, JOHN 98/02/18 13:25:04	SMITH, FRED 98/02/18 13:29:21	COLLINS, MARY 98/02/23 17:01:55	DOE, JOHN 98/02/23 09:00:00	HOWE, TOM 98/02/23 04:14:20	JONES, FRED 98/02/23 11:16:44	COLLINS, MARY 98/03/01 10:15:40
-----------------------------	-------------------------------	---------------------------------	-----------------------------	-----------------------------	-------------------------------	---------------------------------

Output stream

COLLINS, MARY 98/03/01 10:15:40	DOE, JOHN 98/02/23 09:00:00	HOWE, TOM 98/02/23 04:14:20	JONES, FRED 98/02/23 11:16:44	SMITH, FRED 98/02/18 13:29:21
---------------------------------	-----------------------------	-----------------------------	-------------------------------	-------------------------------

Read LOGTIMES file. Delete duplicate records, keeping the last duplicate each time

1.70 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **DELDUPES** stage compares the first line of consecutive messages and deletes consecutive duplicates. The duplicate messages are written to the secondary output stream, if connected. To delete duplicate lines within a multiline message, use the **SEPARATE** stage before the **DELDUPES** stage.

In this example, duplicate records are removed from the LOGTIMES file based on columns 1 to 18 (last name). The first and third records are removed from the output stream. The LOGTIMES file does not change.

COUNT stage: Counting numbers of records

IBM

COUNT stage: Counting numbers of records

Pipe NETV CDRMS | CONS | COUNT **LINES FROM 0**
| COLOR BLUE REV | CONS

```
NetView V6R100          Tivoli NetView    AOFDA NETOP1  06/14/11 14:50:09
* AOFDA PIPE NETV CDRMS | CONS | COUNT LINES FROM 0 | COLOR BLUE REV | CONS
C AOFDA DISPLAY NET,CDRMS,SCOPE=ALL
A AOFDA IST097I DISPLAY ACCEPTED
A AOFDA
IST350I DISPLAY TYPE = CDRMS
IST089I CDRMS TYPE = CDRM SEGMENT
IST089I ADCD6 NEVAC, SA      6, EL   1, NETID = ADCD
IST482I ADCD7 NEVAC, SA      7, EL   1, NETID = ADCD
IST1454I END      2 RESOURCE(S) DISPLAYED
IST314I END
A AOFDA
```

Pipe NETV CDRMS | CONS | COUNT **Messages FROM 0**
| COLOR BLUE REV | CONS

```
NetView V6R100          Tivoli NetView    AOFDA NETOP1  06/14/11 14:53:24
* AOFDA PIPE NETV CDRMS | CONS | COUNT MESSAGES FROM 0 | COLOR BLUE REV | CONS
C AOFDA DISPLAY NET,CDRMS,SCOPE=ALL
A AOFDA IST097I DISPLAY ACCEPTED
A AOFDA
IST350I DISPLAY TYPE = CDRMS
IST089I CDRMS TYPE = CDRM SEGMENT
IST089I ADCD6 NEVAC, SA      6, EL   1, NETID = ADCD
IST482I ADCD7 NEVAC, SA      7, EL   1, NETID = ADCD
IST1454I END      2 RESOURCE(S) DISPLAYED
IST314I END
A AOFDA
```

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **COUNT** stage counts the number of messages, lines, or bytes received on its primary input stream. The count passes to its primary output stream when the input stream disconnects. The first example counts the number of lines on the pipeline (eight). The second example counts the number of messages in the pipeline (three): the **DISPLAY** command, **IST097I**, and **IST350I**. The first **CONS** stage is used for these examples to show the contents of the pipeline.

Student exercise



1-72

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 5.

Lesson 6: Processing pipeline data



Lesson 6: Processing pipeline data

- Discarding pipeline data
- Logging pipeline data
- Saving pipeline data
 - SAFE
 - KEEP

1-73

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Pipeline data can be discarded, displayed, logged, or saved in NetView storage for use later by the same EXEC or a different EXEC. The CONSOLE stage can be used for displaying data to a NetView operator.

HOLE stage: Deleting all data from pipeline



HOLE stage: Deleting all data from pipeline

- Store output of LIST command in stem variable, displaying only text message:

```
/* REXX EXEC */  
'Pipe NETV List Status=Tasks',  
'| STEM MYVAR. | HOLE',  
'| Lit / COMMAND COMPLETE / | CONSOLE'
```

- Test if command generates correlated output:

```
PIPE NETV your_command_name | HOLE
```

If you see output, *your_command_name* is not correlatable

1.74

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **HOLE** stage discards the contents of the pipeline. In the first example, HOLE is used for discarding the data in the pipeline after the data is saved to the stem variable. The LIT stage inserts text into the pipeline to say COMMAND COMPLETE. The second example can be used for determining if a command is *correlatable*, that it has correlated output. If the command response is displayed, it is not correlatable. You can also use the HOLE stage to discard output stream from a multiple-output pipeline.

LOGTO stage: Log messages

IBM

LOGTO stage: Log messages

```
graph TD; LOGTO[LOGTO] --- ALL[ALL]; LOGTO --- HCYLOG[HCYLOG]; LOGTO --- NETLOG[NETLOG]; LOGTO --- SYSLOG[SYSLOG]
```

- LOGto: Send a copy of the pipeline contents to a specified log
The contents remain in pipeline for processing by next stage
- NETLOG: Send a copy of message to NetView DSILOG
- HCYLOG: Send a copy of message to hardcopy log device
- SYSLOG: Send a copy of message to system log

```
Pipe LIT /log this message/ | LOG NETLOG
```

1-75

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **LOGTO** stage sends a copy of the contents of the pipeline to one or more of the specified logs. The input stream (unmodified) passes to the output stream for processing by the next **LOGTO *** stage. Pipeline contents are be logged based on default settings that are defined in CNMSTYLE or with the NetView OVERRIDE and DEFAULT commands.

The default logging settings that you can find in CNMSTYLE are as follows:

```
DEFAULTS.NetLog = Yes  
DEFAULTS.SysLog = No  
DEFAULTS.HcyLog = Yes
```

You can issue a **LIST DEFAULTS** command to display the settings as follows:

DWO654I DISPLAY DEFAULTS
SYSLOG: NO
NETLOG: YES
HCYLOG: YES
And so on

SAFE stage: Storing pipeline data

IBM

SAFE stage: Storing pipeline data

```
.- *----.  
>>-SAFE-----+-----><  
  '- name'   +- APPEND+-  
      '- SEIZE--'
```

SAFE: A place to store one or more messages associated with a procedure, retaining all attributes of the messages

- Procedures can read from and write to the safe place
When writing, the data can be replaced or appended
- Default safe (*) place holds only one message, usually message from automation
- Named safe place can hold multiple messages
Can also be used by nested procedures
- SEIZE reads messages from place and empties it, improving performance
- The safe place empties when procedure ends

1-76

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **SAFE** stage defines a place to store one or more messages *associated with a command procedure*, for example, a REXX EXEC. With the SAFE stage, you can read from or write to a default or a named SAFE.



Note: The messages in a SAFE retain their full message structure and attributes.

When SAFE is the first stage, the specified SAFE is read into the pipeline. For a named SAFE, multiple messages could potentially go into the pipeline.

When SAFE is not the first stage, the input messages are written to the specified SAFE. For the default SAFE, just one message is written and all messages are copied to the output stream. For a named SAFE, each input message is written to the SAFE and to the output stream.

Default SAFE

The default SAFE is the current message that is associated with a command procedure. For example, when the automation table invokes a command procedure, the default safe contains the automated message. The default SAFE is preserved while the command procedure is active. The default SAFE can contain only one message.

Named SAFE

The named SAFE is a named area for a queue of messages associated with a command procedure or group of nested command procedures. For example, a REXX EXEC can write messages to a named SAFE, then call a second EXEC to read from, or write to, the named SAFE.



Note: A named SAFE is preserved while the command procedure or command group is active.

A named SAFE can contain any number of messages. A command procedure group can have any number of named SAFEs at a time.

- APPEND: Specifies that data should be added after data that already exists in the named SAFE. APPEND is valid only when using a named SAFE.
- SEIZE: Use SEIZE for performance improvement when you do not need the contents of the safe to remain in the safe after a read operation.

SAFE stage examples



SAFE stage examples

- Save data in named safe, MYSAFE, for use twice in procedure
 - Save MYSAFE data in a variable
 - Display MYSAFE data on screen

```
/* REXX EXEC */  
'Pipe LIT /Save This/ |  SAFE MYSAFE'  
.  
.  
'Pipe SAFE MYSAFE | VAR text_string'  
'Pipe SAFE MYSAFE | CONSOLE'
```
- Read and empty contents of named safe storing data in common global array for use by other procedures

```
'Pipe SAFE SYSERR SEIZE | STEM (COMMON)  SYSERR. '
```

1.77

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The second SAFE example reads the contents of the named safe, SYSERR, into the common global (stem) variable *syserr*, emptying the contents of the safe.

KEEP stage

IBM

KEEP stage

```
>>>KEEP-- keepname--+-----+-----+----->
      +- timeout+-  +- APPEND+-  

      '- *-----'  '- SEIZE--'  
  
>---+ NOSPILL-----+----->  
  +- SPILL-----+  
  '- ENDCMD /cmd_string/-'
```

KEEP: Defines a task-wide place to store messages, retaining all attributes of the messages

- Procedures can read from and write to the keep place, replacing or appending data
- If no data is written to keep before *timeout* expires, the keep place is purged
Adding data to the keep resets the interval
- SPILL displays contents of the keep place when *timeout* expires
- ENDCMD defines a command to be issued for each message in the keep place when *timeout* expires
- SEIZE reads messages from the keep place and empties it, improving performance

1-78

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **KEEP** stage is similar to the SAFE stage. With the KEEP stage, you can define a task-wide place to store messages. You can read from or write to storage that the *keepname* defines. The name and message are global for the task and exist beyond the life of the procedure that creates them.

If PIPE KEEP is the first stage, it copies messages from the KEEP into the output stream. If PIPE KEEP is *not* the first stage, it copies messages from the input stream into the KEEP.

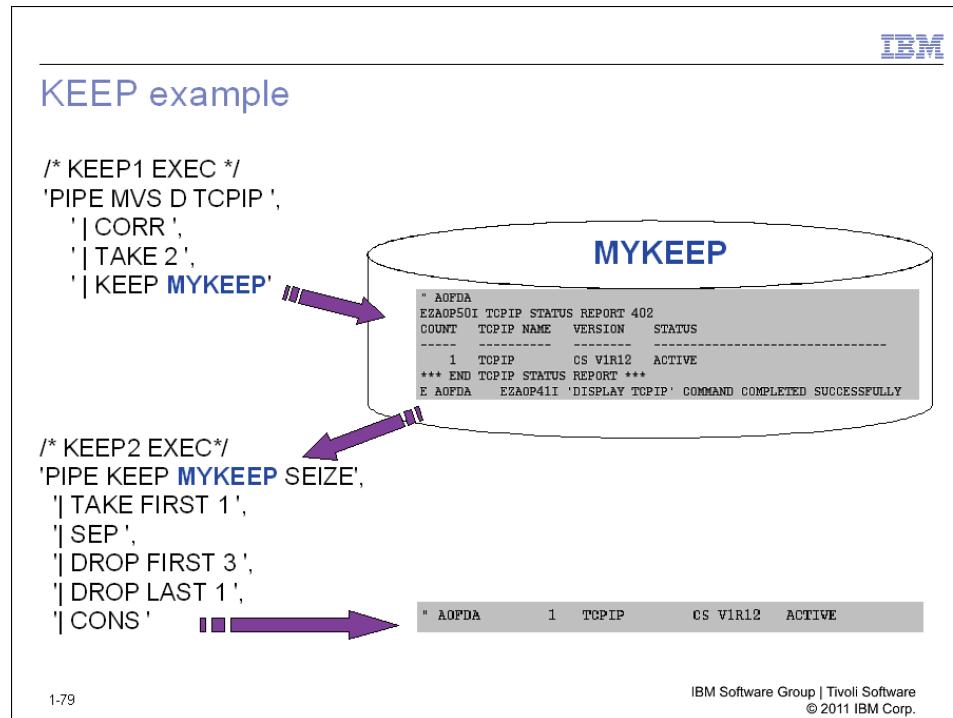
You can use the SPILL operand when KEEP is not a first stage. When the KEEP expires, SPILL indicates to display messages that are in the KEEP. The messages become subject to automation and message traps. If the KEEP expires because of a LOGOFF command or task termination, messages route to the authorized receiver.

You can use the NOSPILL operand when KEEP is not a first stage. When the KEEP expires, NOSPILL indicates to discard messages that are in the KEEP.

Use the **QRYKEEP** command to display keep statistics as follows:

BNH560I KEEP Status for NETOP1					
Keep	Name	Number	Messages	Total Storage	Time Out
KEEP1		40		18212	1000
KEEP2		1		1040	*

KEEP example



This slide shows a simple example of using a KEEP to share data between two REXX EXECs:

1. KEEP1 is a REXX EXEC that issues the MVS D TCPIP command and stores the output in a KEEP called MYKEEP. Immediately after storing the data, KEEP1 ends.
2. EXEC KEEP2 runs. KEEP2 reads the contents of MYKEEP and displays one message on the NetView console. As KEEP2 reads MYKEEP, it deletes the data from the keep with the SEIZE option.

Student exercise

Student exercise



1-80

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 6.

Lesson 7: Advanced topics



Lesson 7: Advanced topics

- Cross-domain pipelines
- Pipe within pipe
- Multiple input and output streams
- Exposing pipeline messages to automation
- Building large pipelines dynamically: INTERPRT stage

1-81

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This lesson discusses advanced pipeline techniques. The INTERPRT stage is similar to a REXX INTERPRT instruction. Using the INTERPRT stage, you can dynamically build and run more complex pipelines.

Cross-domain pipelines

IBM

Cross-domain pipelines

Pipe CORRCMD AOFDB: list dsilog | color blue | lit /Cross-domain PIPE to AOFDB:/ | CONS

```
NetView V6R1M0          Tivoli NetView     AOFDA NETOP1   06/14/11 16:09:04
* AOFDA      PIPE CORRCMD AOFDB: LIST DSILog | COLOR BLUE | LIT /CROSS-DOMAIN
|           PIPE TO AOFDB:/ | CONS
| AOFDA      CROSS-DOMAIN PIPE TO AOFDB:
- AOFDB
TYPE: OPT TASKID: DSILog    TASKNAME: DSILog    STATUS: ACTIVE
MEMBER: DSILogBK
PRIMARY:DSILogP STATUS:ACTIVE    SECONDARY:DSILOGS STATUS:INACTIVE
AUTOFILIP: YES             RESUME: YES
LOADMOD: DSIZDST
Task Serial: 321 REXX Environments: 1 (1%)
Messages Pending: 0 Held: 0
WLM Service Class: Not Available
END OF STATUS DISPLAY
```

Using a labeled RMTCMD: Issue a **LIST DSILog** command in remote domain AOFDB on same operator ID in as AOFDA (NETOP1). Return results in blue as a multiline message. A literal string is also inserted

1-82

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This example shows a labeled RMTCMD using CORRCMD to issue a LIST DSILog in a remote NetView domain. The PIPE is issued from NETOP1 in the AOFDA domain to the AOFDB domain as the same operator. The LIST DSILog response returns as blue text. The LITERAL string remains unchanged because it was inserted after the COLOR stage. The CORRCMD stage includes any pertinent pipeline stages from DSICCDEF.

Pipe within a pipe



Pipe within a pipe

1 Using a labeled RMTCMD: Issue a **PIPE** command to **PING** a resource

```
PIPE CC AOFDB:
PIPE (STAGESEP %) NETV PING 10.44.15.200 % CORR 3
% TAKE LAST 2 % COLOR BLUE REV % CONS ONLY | CONS ONLY
```

```
NetView V6RIMO          Tivoli NetView   AOFDA NETOP1  06/14/11 16:14:40
* AOFDA    PIPE CC AOFDB: PIPE (STAGESEP %) NETV PING 10.1.32.49 % CORR 3 %
  TAKE LAST 2 %COLOR BLUE REV % CONS ONLY | CONS ONLY
C AOFDB    BNH769I 3 packets sent, 3 packets received, 0.00% packet loss
C AOFDB    BNH770I Round trip times from 3 to 7 ms, averaging 4ms
```

2 Take the last 2 lines in the response and return results in reverse-video blue to the outer pipeline

The **PING** command runs in remote domain AOFDB under the same operator ID as in AOFDA (NETOP1)

1-83

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This example shows how to issue a *pipe within a pipe* for sending a command to a cross-domain NetView. A *pipe within a pipe* produces performance while a second system processes data and returns only necessary responses.

A second stage separator character (%) is necessary to distinguish pipeline stages between the inner and outer pipes. Any CORRWAIT value in the inner pipe must be smaller than the CORRWAIT value in the outer pipe.

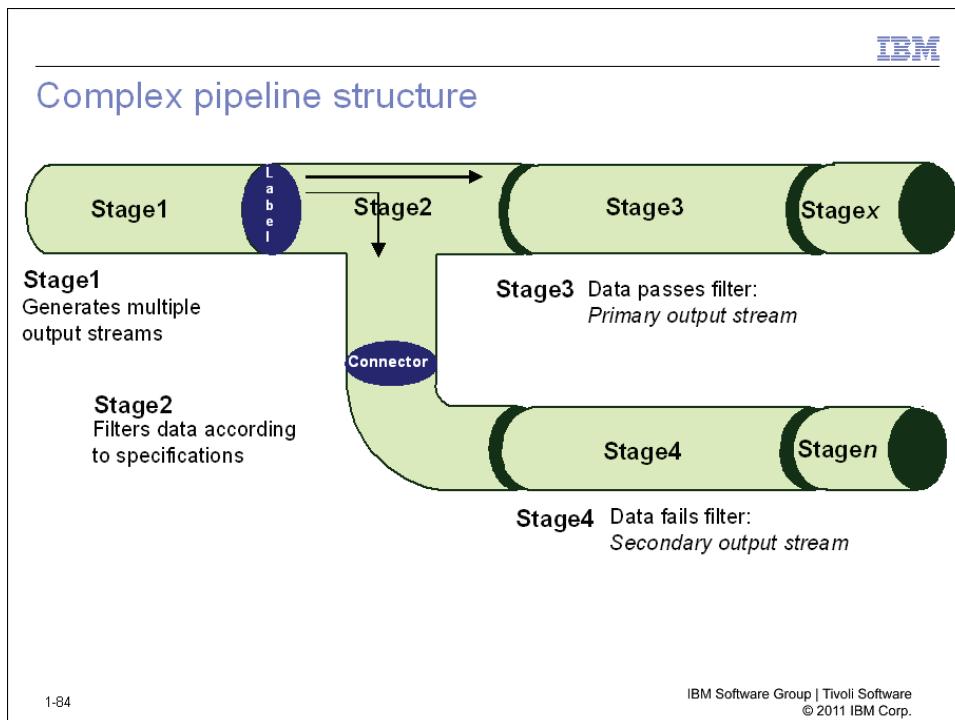
The complete ping response is as follows:

```
BNH765I Pinging tived1.tived.ibm.com at 10.1.32.49 with 3 packets
of length 16 bytes
BNH767I 16 bytes received from 10.1.32.49: seq=1 in 3ms
BNH767I 16 bytes received from 10.1.32.49: seq=2 in 7ms
BNH767I 16 bytes received from 10.1.32.49: seq=3 in 3ms
BNH769I 3 packets sent, 3 packets received, 0.00% packet loss
BNH770I Round trip times from 3 to 7 ms, averaging 4ms
```

This pipeline uses only the last two messages.

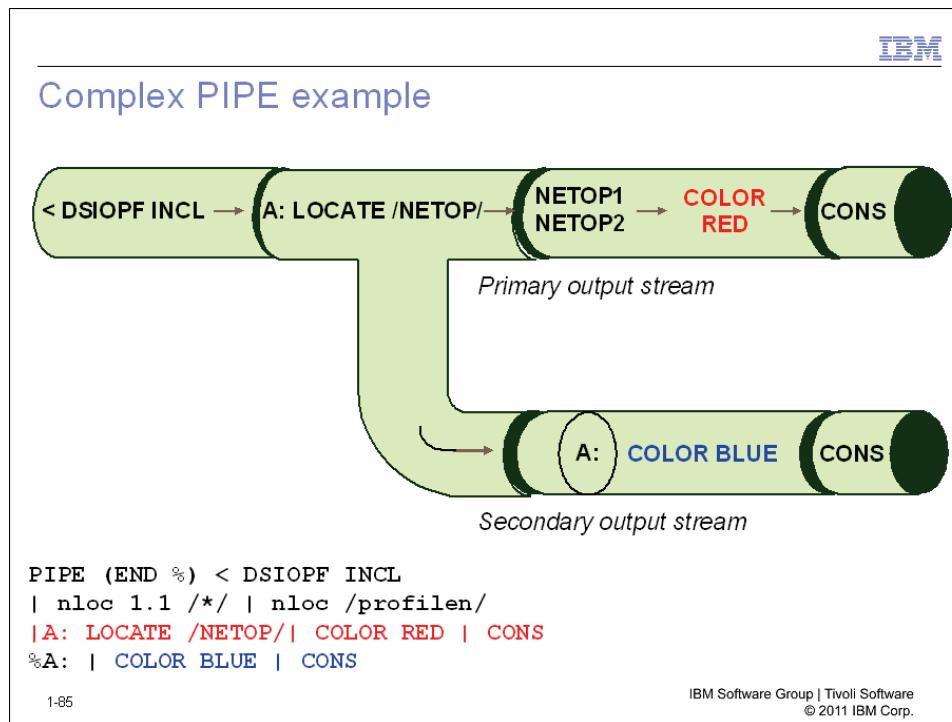
The inner pipeline ends with a CONS ONLY stage. The messages in the output stream flow to the input for the next stage of the outer pipeline, another CONS ONLY stage. CONS ONLY is necessary for both inner and outer pipelines to display the response to the PING command.

Complex pipeline structure



A complex pipeline is made up of several simple pipelines that are connected with labels. Complex pipelines are simple programs rather than complicated commands.

Complex PIPE example



This example reads all lines that are contained in the DSIOPF file (plus included members) into the pipeline. The label **A:** on the LOCATE stage creates a *secondary* output stream for the LOCATE stage. All data in the pipeline that matches the LOCATE stage specification (inclusion of the NETOP character string) passes to the *primary* output stream. All data that does not match passes to the secondary output stream.

The COLOR stage modifies the color attribute of the records that the LOCATE stage selects to red on its primary output stream.

A connector labeled **%A** indicates the end of the first (simple) pipeline that is identified by the label **A:**. It indicates the target of the secondary output stage of the first pipeline LOCATE stage. All data in the secondary output stream is to change its color attribute to blue.

NOT stage: Exchanging input and output streams

IBM

NOT stage: Exchanging input and output streams

Pipe MVS D T | CORR | TAKE 1 | NOT CHOP 8 | CONSOLE

```
NetView V6R1M0      Tivoli NetView    AOFDA NETOP1   06/14/11 16:37:20
* AOFDA    PIPE MVS D T | CORR | TAKE 1 | NOT CHOP 8 | CONSOLE
E AOFDA    LOCAL: TIME=16.37.19 DATE=2011.165 UTC: TIME=15.37.19 DATE=2011.165
```

- The command deletes the first eight characters for each line of the MVS D T command output
- Secondary output stream contains the first eight characters, IEE136I, and a blank

1-86

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can use the **NOT** stage to exchange the input and output streams. CHOP, TAKE, TOSTRING, and LOCATE are some of the stages that can use a NOT stage.

Multiple input and output streams



Multiple input and output streams

- PIPE stages can have multiple input and output streams:
 - FANOUT: Up to ten output streams are supportable
 - FANIN and FANINANY: Up to ten input streams are supportable
- Adjacent stages:
Output stream of first stage becomes input stream of second stage
- Non-adjacent stages:
 - Using label (connector) passes output as input
 - Using END character identifies end of one simple pipeline and start of the next pipeline
- Related PIPE stages:
 - LOOKUP stage compares two input streams
 - PIPEND stage causes a complex pipeline to end immediately

1-87

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

When stages are adjacent each other in a pipeline, the output stream of a stage connects to the input stream of the stage that follows. When stages are *not* adjacent each other in a pipeline, labels and connectors are used for defining the input stream to the non-adjacent stages.

The *end* character indicates the end of one pipeline and the beginning of the next pipeline. The % symbol (END %) is the most common character.

A label is used for creating multiple data streams for a stage. These data streams are numbered, starting with one, and can go as high as 10, depending on the stage. When a stage is processed, the number one (primary) input stream connects to the previous stage if there is one. The number one (primary) output stream connects to the following stage if there is one.

When a connector is encountered later in the pipeline specification, a data stream is defined. The data stream connects from the stage where the label was defined to the connector. The lowest stream number available is assigned to the data stream.

If the labeled stage has an output in a simple pipeline within a complex pipeline, the data stream becomes an output from the stage that defines the label. It also becomes an input to the stage that follows the connector. If the labeled stage is an output to a stage in the pipeline specification, the data stream becomes an input to the stage that defines the label. The data stream also becomes an output to the stage that precedes the connector.

It is possible for a connector to be neither first nor last, in which case, the connector defines both an input and an output for the labeled stage. It is also possible to use two connectors in a row. This usage connects the output of one labeled stage to the input of another.

FANIN, FANINANY, and FANOUT

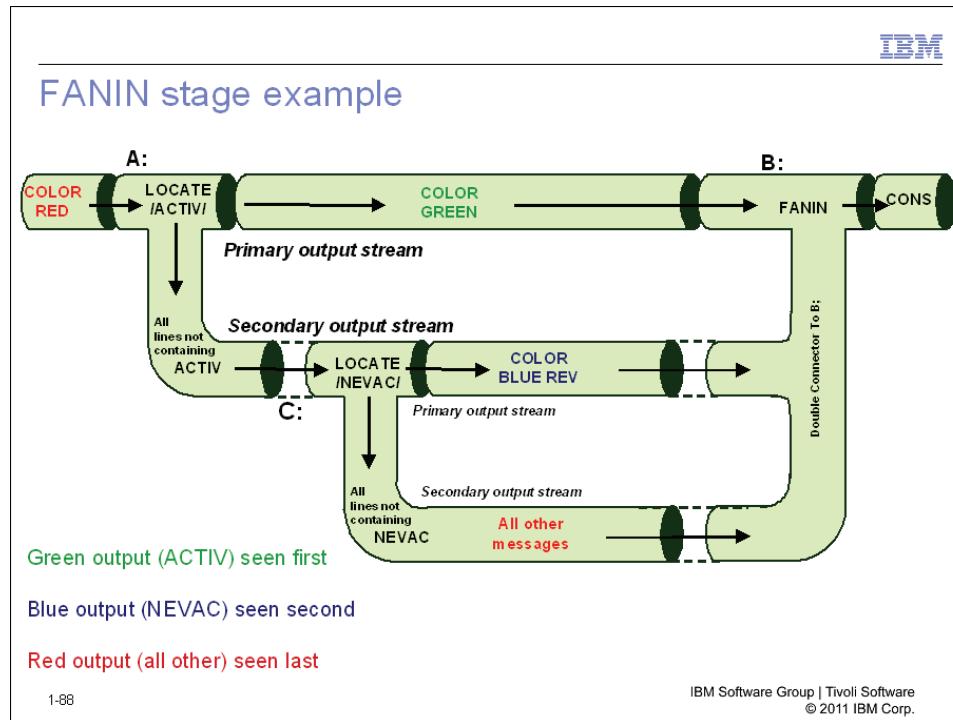
The FANIN stage reads from multiple input streams. FANIN reads from the first stream until that stream disconnects. FANIN then reads from the next input stream until it disconnects, and so on. All data that FANIN reads passes to a single output stream.

The FANINANY stage reads from each connected input stream and passes the messages to a single output stream. Messages pass in the order received regardless of their input stream.

The FANOUT stage copies the messages that it receives on its primary input stream to all connected output streams. Messages that copy to the output streams are identical except for the following cases:

- Messages that are written to the primary output stream retain the IFRAUPRI primary attribute of the original message. The copy attribute, defined by IFRAUCPY, is set to zero.
- Messages that are written to all other output streams have the IFRAUPRI primary attribute setting set to zero and the copy attribute, defined by IFRAUCPY, set to one.

FANIN stage example



This example shows a complex pipeline where the **FANIN** stage collects all the output stages. The REXX program is as follows:

```
*****
/* Example of a complex pipeline using the FANIN stage */
*****
MSG = 'IST482I'          /* */
'PIPE (END +) NETV CDRMS   /* + is the ending character of a pipe */
'| SEPARATE                /* separate to single line messages */
'| LOCATE /'MSG'/           /* locate the IST482I message */
'| CHANGE /'MSG'//          /* discard the MSGID */
'| PRESATTR RED             /* set color red for all lines */
'| A: LOCATE /ACTIV/        /* primary FANIN input stream (ACTIVs) */
'|                   /* pass lines containing ACTIV to primary */
'|                   /* output, others to secondary output A: */
'| PRESATTR GREEN            /* primary output is coloured green */
'| B: FANIN                 /* read all 3 input streams */
'| CONSOLE                  /* output to console */
'|+ A:                      /* connect sec. outp. of LOCATE /ACTIV/ */
'|                   /* to next stage LOCATE /NEVAC/ */
'| C: LOCATE /NEVAC/         /* locate all NEVACs, pass the rest to C: */
'| PRESATTR BLUE REV         /* NEVACs are colored blue */
'| B:                       /* secondary FANIN input stream (NEVACs) */
'|+ C:                      /* connect sec. outp. of LOCATE /NEVAC/ */
'|                   /* to the tertiary FANIN input stream */
'| B:
EXIT
```

All active resources (LOCATE /ACTIV/) are to be displayed in green. All never-active resources (LOCATE /NEVAC/) are to be displayed in blue. All other resources are to be displayed in red.

Student exercise

Student exercise



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 7.

Lesson 8: PIPE EDIT stage



Lesson 8: PIPE EDIT stage

- Modification of pipeline data
 - Create new message
 - Reformat
 - Modify line attributes (for example, color, line type)
 - Perform conversions (for example, SUBSTR, C2X)
 - Dynamically build commands
- PIPE EDIT capabilities
 - Global orders
 - Input orders
 - Output orders
 - Conversion orders

1-90

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

With the **EDIT** stage, you can make many types of changes, or edits, to a message within a pipeline. Edit data can include the following sources:

- Message data
- Line attributes
- Message attributes
- Command responses
- Literal data

With EDIT, you can create or reformat messages. In some cases, you can modify message and line attributes. You can edit as follows:

- Prevent calling REXX programs to manipulate messages.
- Improve performance. Editing within a pipeline flow is faster than driving a command to make the changes.
- Preserve message attributes when changing the message text.
- Improve programmer productivity when writing procedures to manipulate message data.
- Dynamically create commands to issue that are based on the contents of pipeline data.

EDIT is a simple stage consisting of global orders and edit phrases. Edit phrases define the action to be taken on the data as it flows in the pipeline.

- Global orders: Define the overall environment for the subsequent edit phrases. Global orders are optional.
- Input orders: Define the source of data to be processed by the conversion and output orders of the edit phrase.
- Output orders: Define how the resulting data is to be placed in the output line and, subsequently, on the output data stream.
- Conversion orders: Define how the data is to be manipulated. Conversion orders are optional.

PIPE EDIT stage example

IBM

PIPE EDIT stage example

Pipe netv WHO | DROP FIRST 1 | DROP LAST 1 | loc 1.9 /OPERATOR:/ |
EDIT word 1 1 word 2 nw word 5 20 word 6 nw | color blue | coll | cons

```
LIST STATUS=OPS
OPERATOR: NETOP2      TERM: A01A701    STATUS: ACTIVE
OPERATOR: MVS AUTO     TERM: MVS AUTO   STATUS: ACTIVE
OPERATOR: AUTO DVIPA   TERM: AUTO DVIPA STATUS: ACTIVE
OPERATOR: AUTO OAON    TERM: AUTO OAON  STATUS: ACTIVE
OPERATOR: AUTO ODC1    TERM: AUTO ODC1  STATUS: ACTIVE
OPERATOR: AUTO ODC2    TERM: AUTO ODC2  STATUS: ACTIVE
OPERATOR: AUTO ODC3    TERM: AUTO ODC3  STATUS: ACTIVE
...
STATION: NETOP2        TERM: A01A701
HCOPY: NOT ACTIVE      PROFILE: DSIPROFB
STATUS: ACTIVE          IDLE MINUTES: 0
...
END OF STATUS DISPLAY
```

1-91

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Explanation of PIPE EDIT stage in this example is as follows:

1. Issue the NetView WHO command. Delete the first and last lines in the response. Locate only the lines that begin with the string, OPERATOR:.
 2. Use the PIPE EDIT stage:
 - a. WORD 1 1: Take the first word (OPERATOR:) from the input stream, and write it to the first word in *position* one of the output stream.
 - b. WORD 2 NW: Take the second word (for example, NETOP2) from the input stream, and write it to the *next word* of the output stream.
 - c. WORD 5 20: Take the fifth word (STATUS:) from the input stream, and write it to the output stream, beginning in *position* 20.
 - d. WORD 6 NW: Take the sixth word (for example, ACTIVE) from the input stream, and write it to the *next word* of the output stream.
 3. Display the result in blue at the NetView operator screen. The result should be similar to the following text:

```
OPERATOR: NETOP2    STATUS: ACTIVE  
OPERATOR: MVSAUTO   STATUS: ACTIVE  
OPERATOR: AUTDVIPA  STATUS: ACTIVE  
OPERATOR: AUTOAON   STATUS: ACTIVE  
OPERATOR: AUTODC1   STATUS: ACTIVE
```

OPERATOR: AUTODC2 STATUS: ACTIVE
OPERATOR: AUTODC3 STATUS: ACTIVE

And so on.

PIPE EXPOSE stage

IBM

PIPE EXPOSE stage

```
.-- RESPECT--.
|--EXPOSE--+-----+-----+-----+
    +- FORCE---+   '- NOLOG-'
    +- COMMAND---+
    '- TOTRAP-- '
```

- Presents pipeline data to NetView automation:
- NetView exit DSIEX02A
- TRAP and WAIT processing
- Automation table
- NetView exit DSIEX16
- ASSIGN COPY routing
- Logging to the network log, system log, or hardcopy log, as appropriate

1-92 IBM Software Group | Tivoli Software
 © 2011 IBM Corp.

The **EXPOSE** stage causes messages in the pipeline to be exposed to automation by passing the pipeline messages to areas as follows:

- User exit DSIEX02A
- TRAP and WAIT processing
- Automation table for possible automation
- User exit DSIEX16
- ASSIGN COPY routing
- Logging (network log, system log, or hardcopy log)

You can specify actions as follows:

- COMMAND: Specifies that messages that the processing of a command generates in a previous CORRCMD, NETVIEW, or MVS stage are to be exposed. This action is to occur before the messages are absorbed into the pipeline.
- FORCE: Specifies that messages are to be exposed to user exit 02A, message automation, and user exit DSIE16. This action is to occur regardless of whether they have been previously exposed to those interfaces or not.
- NOLOG: Specifies that messages are to be processed as indicated by other specified keywords, but no logging is to occur.
- RESPECT: Specifies that messages that have been exposed to exit 02A, message automation, or exit 16 are not to be exposed to the same interfaces again.
- TOTRAP: Specifies that messages are to be exposed to only TRAP processing. With TOTRAP no logging occurs.

To expose the messages that the MYCLIST EXEC generates for trapping and waiting processing only, the text is as follows:

```
'PIPE NETV MYCLIST | EXPOSE TOTRAP | CONS ONLY'
```

Student exercise

IBM

Student exercise



1-93

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 8.

Lesson 9: Full-screen automation



Lesson 9: Full-screen automation

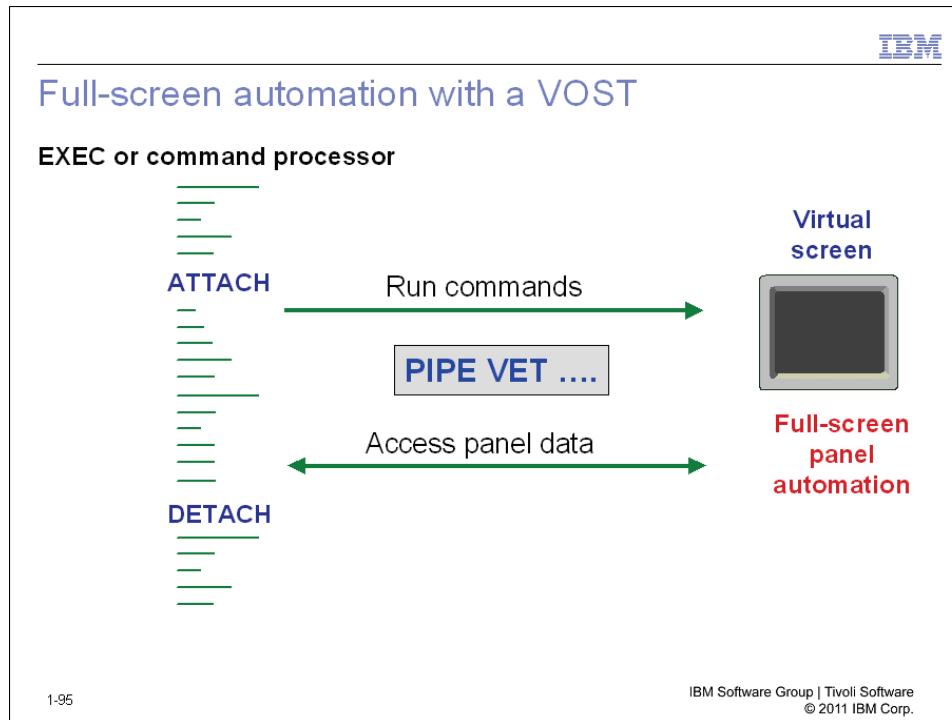
- Enables a program to interact with full-screen applications:
 - REXX, PL/I, or C
 - Simulation of an operator
- The program can perform as follows:
 - Reads data from an application panel
 - Writes data to an application panel
 - Simulates pressing keys on application panel
 - All PF keys, PA keys, Enter, or Clear

1-94

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Using a REXX, PL/I, or C program, you can connect to a full-screen application as a simulated operator. Under the simulated operator, you can issue commands and interrogate the screen for a response. This type of automation is typically called *screen scraping*. You can use a TAF full-screen session to connect to the application.

Full-screen automation with a VOST



This slide provides an overview of the basic steps to implement full-screen automation.

- The **ATTACH** command is used for beginning a simulated operator session, called a virtual OST (VOST), and to issue a command in that session. The virtual screen that is created on the VOST is 24 rows by 80 characters with no query support.
- The **PIPE VET** stage is used for reading data from and writing data to a virtual screen that is associated with a virtual OST (VOST). Row and field form data returns in a multiline message, BNH150I. NetView also provides a VET command.
- The **DETACH** command is used for ending the virtual OST (VOST) that an ATTACH command creates. DETACH simulates sending a LOGOFF command to the application that runs on the VOST.

See the *IBM Tivoli NetView for z/OS 6.1 Programming: Pipes* manual for more details. Online help is also available for ATTACH, DETACH, and PIPE VET commands.

NetView also provides examples of coding full-screen automation. Browse CNMS1101 in CNMSAMP or CNME2011 in CNMCLST to see sample REXX code.

Student exercise

Student exercise



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 9.

Summary



Summary

Now that you have completed this unit, you can perform the following tasks:

- Discuss the basics of NetView PIPEs
- Use NetView PIPEs to perform the following tasks:
 - Issue commands
 - Trap and parse messages
 - Set and retrieve global variables
 - Read from and write to files
 - Perform automation

1-97

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

IBM Tivoli certification and training

In today's global business world, enhancing and maintaining skills is essential to keeping pace with rapidly changing technologies. Businesses need to maximize technology potential and employees need to keep up to date with the latest information. Training and professional certification are two powerful solutions.

Certification

There are many reasons for certification:

- You demonstrate value to your customer through increased overall performance with shorter time cycles to deliver applications.
- Technical certifications assist technical professionals to obtain more visibility to potential customers.
- You differentiate your skills and knowledge from other professionals and stand out as the committed technical professional in today's competitive global world.

Online certification paths are available to guide you through the process for achieving certification in many IBM Tivoli areas. See ibm.com/tivoli/education for more information.

Special offer for having taken this course

Now through 31 December 2011: For having completed this course, you are entitled to a 15% discount on your next examination at any Thomson Prometric testing center worldwide. Use this special promotion code when registering online or by telephone to receive the discount: **15CSWR**. (This offer might be withdrawn. Check with the testing center as described later in this section.)

Role-based certification

All IBM certifications are based on job roles. They focus on a job a person must do with a product, not just the product's features and functions. Tivoli Professional Certification uses the following job roles used:

- IBM Certified Advanced Deployment Professional
- IBM Certified Deployment Professional
- IBM Certified Administrator
- IBM Certified Solution Advisor
- IBM Certified Specialist
- IBM Certified Operator

Training

A broad spectrum of courses, delivery options, and tools helps keep your employees up to date with the latest IBM Tivoli information:

- *Instructor-led training (ILT)*
Live interaction with an IBM instructor, hands-on lab exercises, and networking with your peers from other companies
ibm.com/tivoli/education
- *Instructor-led online (ILO)*
All the benefits of ILT, but savings on travel dollars and training costs
ibm.com/training/ilo
- *Self-paced virtual classes (SPVC)*
Interactive and hands-on exercises on your schedule
ibm.com/training/us/spvc
- *Web-based training (WBT)*
Training anywhere, any time, that saves you money and travel
ibm.com/training/us/tivoli/wbt
- *Multimedia library*
Modules supporting new and experienced learners with fully animated multimedia clips, step-by-step audio, and companion text
ibm.com/software/tivoli/education/multimedialibrary
- *IBM Education Assistant*
More specific, granular web-based training with individual presentations on specific topics
www-01.ibm.com/software/info/education/assistant/
- *Corporate Education Licensing Program (CELP)*
Solutions for large IBM customers who need to adopt IBM Tivoli's tools and technologies
ibm.com/training/us/tivoli/celp
- *Tivoli training paths*
Course maps with flow charts and course descriptions to help you find the right course
[ibm.com/training/us/tivoli\(paths](http://ibm.com/training/us/tivoli(paths)