



IBM Tivoli NetView for z/OS 6.1: Fundamentals

Student Exercises

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

|||||

Introduction	1-1
Exercise 1-1: The NetView procedures	1-2
Exercise 1-2: CNMSTYLE	1-8
Exercise 1-3: CNMSTYLE Report CNMSJCRG	1-10
Exercise 1-4: IPTRACE and IPSTAT	1-12

Introduction	2-1
Exercise 2-1: Launching the Tivoli Enterprise Portal client in desktop mode	2-2
Exercise 2-2: Using the online documentation and help	2-3
Exercise 2-3: Navigating workspaces	2-4
Accessing default workspaces	2-4
Navigating workspaces by using links	2-6
Accessing other workspaces that a Navigator item associates with	2-7
Exercise 2-4: Working with NetView Health	2-8
Changing Situation values	2-8
Using Expert Advice	2-9
Issuing a command and viewing the command response	2-10
Exercise 2-5: Refreshing workspaces	2-11
Manually refreshing workspaces	2-11
Automatically refreshing workspaces	2-11
Exercise 2-6: Using a 3270 view	2-12
Exercise 2-7: Queries	2-15
Accessing queries	2-15
Using thresholds	2-17
Exercise 2-8: Other Tivoli Enterprise Portal-related functions (optional)	2-19
Saving workspaces	2-19
Accessing workspaces as web addresses	2-19
Modifying view panes	2-20
Resizing views	2-20
Modifying table views	2-21
Resizing columns	2-22
Sorting columns	2-22
Changing column sequence	2-24

Student exercises for Unit 1

.....

Introduction

These exercises provide you with experience using IBM Tivoli NetView for z/OS. After completing this unit, you should be able to perform the following tasks:

- Access the NetView 3270 interface.
- View procedures used to run NetView.
- Work with the CNMSTYLE start-up statements.
- Run a report analyzing the CNMSTYLE start-up statements.

Whenever you are unsure of what you see, access the online help and try to discover the answer before you ask the instructor.

For these student exercises, you use the NetView 3270 interface and the IBM Time Sharing Option (TSO) Interactive System Productivity Facility (ISPF) program.

Your instructor provides the following information to use in these exercises:

Connecting to z/OS system _____
Connecting to VTAM _____
NetView VTAM application name _____
Second NetView VTAM application name _____
NetView operator ID and password _____
Second NetView operator ID and password _____
TSO user ID and password _____
Team DSIPARM data set _____
Team CNMSJCRG report data set _____
JCL PROCLIB name _____
NetView application PROC name _____
NetView SSI PROC name _____
Held MSGCLASS used for retrieving and viewing job output _____

Exercise 1-1: The NetView procedures

Perform the following steps:

- ___ 1. Connect to your z/OS operating system and access a VTAM screen. Your instructor provide instructions for this.
- ___ 2. From the VTAM screen, log on to the NetView application program, and issue the following command:

```
logon applid=netviewapplicationname
```

- ___ 3. Log on with the NetView operator identification (ID) and password that is assigned to your team.

```

NN    NN                      VV      VV
NNN  NN  EEEEE  TTTTTTT  VV      VV  II  EEEEE  WW      WW  TM
NNNN  NN  EE      TT      VV      VV  II  EE      WW      W  WW
NN NN NN  EEEE      TT      VV      VV  II  EEEE      WW  WWW  WW
NN  NNNN  EE      TT      VV  VV      II  EE      WWW  WWW
NN  NNN  EEEEE      TT      VVV      II  EEEEE      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 V6R1M0

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

```

When you first log on, the NetView News displays. Three asterisks at the bottom of the screen means more data is viewable. Press Enter to display the next page. After the News displays content, the main NetView menu opens.

- ___ 4. If you need to find the menu later, return to it by issuing the following command:

mainmenu

You can access many NetView functions from the menu. To exit this screen, press PF3.

```

CNM1NETV          Tivoli NetView for z/OS Version 6 Release 1          Main Menu

                  Operator ID = NETOP1    Application = AOFDA000

Enter a command (shown highlighted or in white) and press Enter.

Browse Facility          BROWSE command
Command Facility        NCCF command
News                    NEWS command
PF Key Settings         DISPFK command
Help Facility           HELP command
Index of help topics    INDEX command
Help Desk               HELPDESK command
Hardware Monitor        NPDA command
Session Monitor         NLDM command
Status Monitor          STATMON command
  
```

- ___ 5. The LISTA command is the next command that you use. To learn about LISTA, issue the following command:

HELP LISTA

To scroll forward in the help screen, press PF8. To scroll backward, press PF7. To exit, press PF3. You use these three program function keys again. To view all of the key settings, issue the following command:

DISPFK

Press PF3 three times to exit the DISPFK, the HELP LISTA, and the main menu.

- ___ 6. In the NetView application, you can use one or more data sets for the initialization. These data sets are partitioned data sets (PDS), allocated to the DSIPARM data definition name

(DDNAME). To view the data sets that is being used by this NetView application, issue the following command:

```
LISTA DSIPARM
```

NetView V6R1M0 Tivoli NetView AOFDA NETOP1 07/01/11 17:16:22			
* AOFDA LISTA DSIPARM			
' AOFDA			
CNM299I			
DDNAME	DATA SET NAME	MEMBER	DISP

DSIPARM	NV390.V6R1M0.WORKSHOP.DSIPARM		SHR, KEEP
	NV390.V6R1M0.USER.DSIPARM		SHR, KEEP
	NV390.DSIPARM		SHR, KEEP
	NV390.SAQNPARM		SHR, KEEP

The DSIPARM for your NetView application program has several data sets that are allocated and concatenated. In a later exercise, you make changes into this data set. Look at the list to ensure that you have one that is named for your team.

7. Some commands might have a lengthy display, resulting in wrapping or requiring you to press the Enter or CTRL key to continue the display. Issue the following command as an example:

```
D NET,APPLS
```

When it is more convenient to scroll command output than to display it in a window, issue the following command:

```
WINDOW D NET,APPLS
```

Press PF8 to scroll forward and PF7 to scroll backward. Another useful command to issue from the CMD==> prompt is FIND. Select some text to look for with the FIND command. Press PF3 to close this screen.

8. Another useful NetView command is BROWSE. Two started tasks make up a typical NetView system. One is the subsystem interface. In your system, it is called AUTOSSI. Ask your instructor for the name of the PROCLIB data set where the NetView procedures are. Issue the following BROWSE command to view the contents of the NetView subsystem interface procedure (proc).

```
BROWSE 'proclibdatasetname(netviewssiproc)'
```

Examine the contents of the sub-system interface proc. You can scroll forward or backward by pressing PF8 and PF7. The procedure has documentation comments for its use. To exit the BROWSE command, press PF3.

9. The second started task in a typical NetView system is the NetView application program. View the contents of this procedure by issuing the following command:

```
BROWSE 'proclibdatasetname(netviewapplicationproc)'
```

Data sets are typically grouped together by the NetView function that is supported. The NetView logs contain information that can be useful in debugging.

- ___ 10. To view the logs by using several methods, issue the following command:

BLOG

The following screen opens.

```

CNMKELIP                               NetView Log Browse                               07/01/11

Display NetView log records for:

  NetView Domain ===> AOFDA              { NetView Netid ===> *          }
                                           { RMTCMD Operid ===> *          }
  NetView Log      ===> NETLOGA

Selection Criteria:

  Display column ===> U17

  From:  Time      ===> [REDACTED]        { Date ===> [REDACTED]        }
  To:    Time      ===> [REDACTED]        { Date ===> [REDACTED]        }

  Operator id      ===> [REDACTED]        { The * and ? wildcards can be used }
  Domain id        ===> [REDACTED]        { anywhere in this group of fields. }
  Message id       ===> [REDACTED]
  Message text     ===> [REDACTED]

CMD==>
TO SEE YOUR KEY SETTINGS, ENTER 'DISPFK'
  
```

- ___ 11. Accept the defaults and press Enter to view the NETLOGA file. You can specify a different log file name on this screen. Press PF7 and PF8 to scroll backward and forward in the log. Press PF3 twice to exit the screen. The second way to browse the NETLOGA file is issuing the two following commands:

BLOG NETLOGA

BROWSE NETLOGA

Yet another way to browse the NetView log and the MVS system log is to use the CANZLOG command, which stands for consolidated audit, NetView, and z/OS log. Browsing both network and system messages in one place can sometimes be very helpful. Enter CANZLOG.

- ___ 12. Accept the defaults and press Enter to view the combined log of NetView and z/OS.

CANZLOG is archived by default, and this is managed by settings in CNMSTYLE.

- ___ 13. Position the cursor on a message in the log and press Enter. The result is a window with attributes for that specific message. To view the complete message, press PF2. Press PF3 to return to CANZLOG, and press PF3 to exit CANZLOG.

14. Using the MVS command, you can issue a command to the z/OS operating system and receive the results on your NetView 3270 session. Issue the following command to view the time of day:

```
MVS D T
```

The command results are immediate. Next, use an automation command.

The AFTER command accepts a timing operand to cause it to wait before issuing the command. Issue this variation of the previous command, which waits for fifteen seconds:

```
AFTER 00:00:15 MVS D T
```

Other useful commands that are used in automation are CHRON, AT, and EVERY.

15. The RMTCMD command is used for issuing a command from one NetView logged-on operator to a second logged-on operator. Log on to NetView on each of the two z/OS systems that is assigned to your team, using two different NetView IDs. Issue the following command from the first system and ID:

```
MVS D A,L
```

From the second NetView domain and operator ID, issue a command like the following example:

```
RMTCMD SEND DOMAIN=netviewdomain,OPERID=secondnetviewoper,MVS D A,L
```

The output should appear similar to the following screen capture, and the output from each of the two z/OS systems should appear slightly different:

```
NetView V6R1M0          Tivoli NetView  AOFDB TS0CW01 07/04/11 16:52:42
* AOFDB  RMTCMD SEND DOMAIN=AOFDA,OPERID=TS0CW02,MVS D A,L
* AOFDA  MVS D A,L
" AOFDA
IEE114I 16.52.42 2011.185 ACTIVITY 315
JOBS      M/S      TS USERS      SYSAS      INITS      ACTIVE/MAX VTAM      ORS
00001     00016     00000     00031     00006     00000/00025     00009
CANACN    CANACN    CNDL     NSW S     LLA      LLA      LLA      NSW S
VLF       VLF       VLF      NSW S     JES2     JES2     IEFPROC  NSW S
NET       NET       VTAM     NSW S     RACF     RACF     RACF     NSW S
RMF       RMF       IEFPROC  NSW S     TSO      TSO      STEP1    OWT S
SDSF      SDSF      SDSF     NSW S     TCPIP    TCPIP    TCPIP    NSW SO
RMFGAT    RMFGAT    IEFPROC  NSW SO    TELNET   TELNET   TN3270   NSW SO
OSNMED    OSNMED    OSNMED   OWT SO    SNMPQE   SNMPQE   SNMPQE   OWT SO
FTPSEVERE STEP1    STCAPP    OWT AO    AUTOSSI  AUTOSSI  NETVIEW  NSW S
AUTONETV  AUTONETV  NETVIEW  NSW SO
```

16. Some TCP/IP commands are available in NetView. Two common commands are PING and TRACERTE. Issue the following PING command with the host name or IP address of a z/OS system that your instructor provides:

```
PING hostnameoripaddress
```

Issue the following trace route (TRACERTE) command with the same host:

`TRACERTE hostnameoripaddress`

- ___ 17. Issue the following BOOKS command to list the web addresses for accessing the publications:

`BOOKS`

Exercise 1-2: CNMSTYLE

Perform the following steps:

- ___ 1. The BROWSE command is very versatile. Log on to your NetView application 3270 interface. Issue the following command to see the options available with the BROWSE command:

```
HELP BROWSE
```

- ___ 2. CNMSTYLE members can cause other DSIPARM members to be loaded at NetView Initialization. This is the first of two ways that you browse the CNMSTYLE settings. Issue the following command to find the data set where the CNMSTYLE member is:

```
LISTA DSIPARM CNMSTYLE
```

Issue the following command to browse the data set and member:

```
BROWSE 'CNM.DSIPARM(CNMSTYLE)'
```

- ___ 3. Use %INCLUDE statements in CNMSTYLE for loading other DSIPARM members. When the CNMSTYLE member is being browsed by the full data set name and member, you can see the %INCLUDE statements. Issue the following command:

```
FIND %INCLUDE
```

You can look for statements where %INCLUDE begins in the first column. If necessary, use the PF5 key as a repeat find key.

- ___ 4. Try a second way to view the CNMSTYLE settings. The help for the BROWSE command illustrates other usable functions. Using another variation, you can view all of the key initialization statements for starting NetView. Specify the member name with the following command:

```
BROWSE CNMSTYLE
```

Issuing the command results in the %INCLUDE statements being resolved and those members also being included in the view.

- ___ 5. Find DATASET: at the top of the view. A number following that text represents the data set number in the concatenation where the member and parameter statements were obtained. Issue the following commands.

```
AUTOWRAP OFF
LISTA DSIPARM
```

Viewing the list of displayed data set names, count from the first number down to the number that you recorded in this step. This is the data set that contains the member that you were browsing. When you finish counting, three asterisks appear at the bottom of the screen. Press Enter to return to the browse view.

- ___ 6. Find the start of the member CNMSTGEN. Issue following command:

```
FIND CNMSTGEN
```

A title displays START OF MEMBER CNMSTGEN FROM CNMSTYLE. The data set number is not normally in the same place as CNMSTYLE. When you browse

CNMSTYLE by the member name only, you do not see the actual %INCLUDE statements. But you do see the contents of that %INCLUDED member.

Other components and products load during NetView initialization with the use of TOWER statements in CNMSTYLE members.

- ___ 7. When browsing the CNMSTYLE member, move to the top of the member by using the **TOP** command. Search for the first occurrence of TOWER in the first column:

```
FIND TOWER
```

A list of tower names follows the TOWER = statement. If an asterisk precedes a tower, the tower is not to be activated or enabled at NetView initialization.

- ___ 8. Some towers have subtowers that are enabled or disabled when NetView starts. These statements contain the name of the tower, a period, the name of the subtower, an equal sign, and the name of the value. Browse CNMSTYLE and find the subtowers for the Tivoli Enterprise Monitoring Agent tower by issuing the following command:

```
FIND 'TOWER.TEMA'
```

You can issue the following command to determine if TEMA subtowers are active on the running system:

```
NACTL LISTINFO
```

- ___ 9. Changes that users make should not be made in CNMSTYLE. One of the %INCLUDE statements causes CNMSTGEN to load. This is the member intended to contain customizations. When NetView loads parameters from the various included members, it can encounter a repeated statement. If it does, the last occurrence is used. Compare changes on this NetView system to originally provided material.
- ___ 10. A default logon profile can be defined in CNMSTYLE for userids not defined to NetView but defined in SAF. Logon to NetView domain AOFDA with user TSCCW20, the result is a normal logon with profile DSIPROFB.

Browse DSIOPF. TSCCW20 is not defined as an operator.

Browse CNMSTGEN to find the default logon profile defined as follows:

```
DEFAULTS.LogProf = DSIPROFB
```

This is the how to define the default logon profile. The default value is *NONE*.

Exercise 1-3: CNMSTYLE Report CNMSJCRG

Perform the following steps:

- ___ 1. Log on to TSO by following the instructions that your instructor provides you.
Because several members are processed for the CNMSTYLE initialization, you can run a batch job for producing a report that shows all of the CNMSTYLE statements. This report shows the CNMSTYLE statements that are used and the members that the statements came from.
- ___ 2. From ISPF, copy member CNMSJCRG from NETV610.CNMSAMP data set to the JCL data set assigned to your team, and edit the member.
- ___ 3. Change the job name on the job card to contain your user ID.
- ___ 4. Add the NOTIFY parameter with your user ID. If your system contains a held MSGCLASS, change the MSGCLASS to that class.

Two DDNAMEs are important for running this job. The first is DSIPARM. This DDNAME must match the name that is used in the NetView application DSIPARM on your z/OS system. The reason is that the loading order of members must be the same for the report to be accurate.

One of the data sets in both this job and the NetView application procedure is a DSIPARM library that is designated for the workshop. This library is normally very early in the concatenation.

The second DDNAME that is important is DSIWRIT. The output from the CNMSJCRG job is directed into that PDS. The member name is CNMCRG.
- ___ 5. Make any necessary changes to the DSIPARM and DSIWRIT DDNAMEs and submit the CNMSJCRG job.
- ___ 6. Use System Display and Search Facility (SDSF) to view the output of the job. From SDSF, issue the ST command to view the jobs that have run pertaining to your TSO user ID. Select the job that ran for CNMSJCRG. Ensure that the return code is 0. If not, correct the problem and re-run the job.
- ___ 7. Using NetView, browse the member CNMCRG from the report PDS assigned to your team. The member CNMCRG contains the output of the CNMSTYLE report.
- ___ 8. Four reports are in CNMCRG. The first report lists information about the CNMSTYLE including order and active towers. Also, the first report includes the general NetView statements. Issue the following command to see where this report starts:


```
FIND 'Statements for function: NetView General'
```
- ___ 9. Several CNMSTYLE members have the same statements that are defined with different assigned values. If more than one member has the same statement, the value that is used is from the last member.

Issue the following command to find the DOMAIN statement:

```
FIND 'DOMAIN ='
```

What was the original value for DOMAIN? _____

Which member was the original member defined in? _____

Which data set contains this member? _____

What is the changed value for DOMAIN? _____

Which member is this new value defined in? _____

Which data set contains this member? _____

(The answers are not provided, as these can be different on your system.)

- ___ 10. Repeat this process for the TOWER statement. The TEMA must be enabled for a later exercise to work correctly.

The second report lists statements that pertain to the various functions and Towers. Each tower, such as SA, AON, and MSM, have their own section. Find any TOWER.TEMA statements. The TEMA subtowers must be enabled for a later exercise to work correctly.

- ___ 11. The third report lists statements used for two types of automated commands. Find where *auxInitCmd statements* begins. This is the first kind of automated command.

The fourth and last report lists the data REXX statements that are used in CNMSTYLE members.

- ___ 12. To run the NetView EMA, two additional started tasks must be active. From NetView, issue the following command to see jobs, started tasks, and TSO users that are active:

```
MVS D A,L
```

CANSDDSST is the default started task name that operates the Tivoli Enterprise Management System database. On our system, we have changed it to CANADSST. If CANADSST is not active, issue the following command:

```
MVS S CANADSST
```

CANSNA is the default started task nTivoli Enterprise Monitoring Serverame that runs the NetView EMA data collectors. On our system, it is changed to CANANA.If CANANA is not active, issue the following command:

```
MVS S CANANA
```

If you want NetView to start the Tivoli Enterprise Monitoring Server and EMA, use additional EMAAUTO statements in CNMSTYLE to accomplish that. The Tivoli Enterprise Monitoring Server and EMA can automatically be started from the *COMMNDxx* member in PARMLIB. NetView automation issues the NACMD command when the agent starts.

With the NetView EMA now running, you access data from the NetView EMA in a later exercise.

Exercise 1-4: IPTRACE and IPSTAT

This exercise familiarizes you with IP management in NetView for z/OS. Perform the following steps:

- ___ 1. Log on to the master domain with the NetView operator identification (ID) and password that is assigned to your team.
- ___ 2. Issue the IPTRACE command, select PKTTRACE, and press Enter.

```
FKJK2A22      PKTTRACE  Control  SYSTCPDA  NONE  for NVDomain: LOCAL

Service Point/Stack: MVSCF01  TCPNAME: TCPIP
PKTS: ACTIVE  On Task: AUTOPKTS  GTF: NO

Start Time:                               _ Start Writer
                                              Writer: PKTCP

Options:                                3-VIEW PACKETS

Infc/Link      Stat Prot IP Address/Prefix      Ports      Record
Src Portnum Dest Count

ALL            *      *                        *      *      *
```

EZL481I ONE OR MORE REQUESTS FAILED, VIEW THE RESPONSE WINDOW OR DSILOG FOR DET
Command ==>

F1=Help F2=Main Menu F3=Return F4=Start SYSTCPDA F5=Refresh F6=Roll
F7=Backward F8=Forward F9=Assist F10=PKTS Management F12=Cancel

- ___ 3. Note that the component SYSTCPDA is not active, marked red at the top. Start SYSTCPDA by selecting ALL and press F4 (Start SYSTCPDA). Wait for the red SYSTCPDA to become green.

```
FKXK2A22      PKTTRACE  Control  SYSTCPDA  ACTIVE  for NVDomain: LOCAL

Service Point/Stack: MVSCF01  TCPNAME: TCPIP
PKTS: ACTIVE  On Task: AUTOPKTS      GTF: NO

Start Time: 2011-07-05-09:22:02          Writer:  *NONE*

Options: 1-START/ADD 2-STOP  3-VIEW PACKETS

  Infrc/Link      Stat Prot IP Address/Prefix      Ports      Record
              Src  Portnm Dest  Count
  _  ALL          *    * _____  *    *    *

EZL481I ONE OR MORE REQUESTS FAILED, VIEW THE RESPONSE WINDOW OR DSILOG FOR DET
Command ==>
F1=Help      F2=Main Menu  F3=Return  F4=Stop SYSTCPDA  F5=Refresh  F6=Roll
F7=Backward  F8=Forward   F9=Assist  F10=PKTS Management  F12=Cancel
```

Exit by pressing F3 two times.

- ___ 4. Enter IPSTAT

On the panel that is presented, enter an asterisk for Hostname and select MVSCFxx.

```
FKXK2200          TCP/IP for z/OS Connection Status          AOFDA

Enter TCP/IP address or HOSTNAME:
*
-----
Service Point  System  IP Address  Host Name
s MVSCF01      TCPIP   10.31.187.194  MVSCF01.ILSVPN.IBM.COM

Command ==> █
F1=Help      F2=Main Menu  F3=Return          F6=Roll
F7=Backward  F8=Forward   F9=Filters        F10=Details       F12=Cancel
```

___ 5. Press Enter.

The next screen lists some active TCPIP sessions for your host.

```
FKKK2210          TCP/IP for z/OS Connection Management

CLIENT
*

Service           Active      IP
Point    Hostname  Connections Address
MVSCF01  MVSCF01.ILSVPN.IB 3      10.31.187.194

Command ==>
F1=Help      F2=Main Menu  F3=Return    F4=Commands  F5=Refresh   F6=Roll
F7=Backward  F8=Forward    F9=Filters   F11=Zoom     F12=Cancel
```


- ___ 7. Position your cursor on the first client and press F4 for commands.

```
FKXK2221      TCP/IP for z/OS Connection Management
CLIENT ----- > Service Point ----- > CONNECTION
                MVSCF01                    39632
10.4.127.184   10.31.187.194                23
.....
Client : IP Address      10.4.127.184      :
Port   : Port           39632              :
39632  : Connection ID   00000541          :
4022   : LU              ESIP1014          :
39631  : APPL            AOFDA019          :
        : Send           163115            :
        : Receive        2319              :
        : Send Window    64128             :
        : Conn ResourceName TELNET          :
        :                                                         :
        :                                                         :
        :                                                         :
        : F1=Help   F4=LU Cnds   F5=APPL Cnds   F6=Roll   F12=Cancel :
        :.....:
Command ==>
F1=Help      F2=Main Menu   F3=Return                      F6=Roll
F7=Backward  F8=Forward     F9=Filters                      F12=Cancel
```

On this panel, you can issue several commands in context.

- ___ 8. Enter “11 Packet Trace” and press Enter.

```

FKXK2A24          Display Packet Control          LOCAL
Service Point/Stack: MVSCF01  Proc: TCPIP    Infc Name: ALL

LAddr 10.31.187.194
RAddr 10.4.127.184

Portnum: *      LPort: 23      RPort: 39632  Protocol ALL (default)
                                           S TCP
                                           - UDP
Time: Start *                                     - ICMP
End *                                     - OSPF
                                           -        (Number)

MaxRecs: 1 1-Last 100      Truncate: 65535
          2-First

Command ==> █

F1=Help          F3=Return    F4=View Packets    F6=Roll
F8=Extended Options  F10=Analyze      F12=Cancel
  
```

Pressing F10 (Analyze) opens a panel with statistics on error flags. If additional errors are displayed, information is available.

___ 9. To view the packets being traced, press F4.

```
FKXK2A26      PKTTRACE  SUMMARY                                     AOFDA
                                                    More:+
DP   Nr hh:mm:ss.rrrrrrrrrr IpId   Seq_num   Ack_num   Wndw Flags
OT10674 09:45:36.474103 638F 2449425877 2451929057 4095 ACK PSH
OT10673 09:45:36.107483 638E 2449424612 2451929049 4095 ACK PSH
          00000000 114F6D29 *.....|_.....Om)*
OT10672 09:45:36.107480 638D 2449423244 2451929049 4095 ACK
          00000200 83F5c211 *....c5B.....*
IT10671 09:45:36.007899 2794 2451929034 2449423244 501 ACK PSH
          00000000 547D5A5E *.....'!; ....T}Z.*
OT10654 09:44:24.377734 6385 2449423244 2451929034 4095 ACK PSH
IT10653 09:44:24.077423 2793 2451929026 2449423244 501 ACK PSH
          02000000 8200FFEF *....b... .....*
IT10652 09:44:24.075439 2792 2451929026 2449423244 491 ACK
IT10651 09:44:24.073734 2791 2451929026 2449423039 492 ACK

Command ==>
F1=Help          F3=Return      F4=Details    F5=Refresh    F6=Roll
F7=Backward F8=Forward    F9=Commands   F11=Right     F12=Cancel
```

___ 10. Position the cursor on a packet, and press F4 (Details).

___ 11. Exit by pressing F3 several times, answer Y to the question if trace should be stopped.

You should now be familiar with accessing TCP/IP session and tracing data.

Student exercises for Unit 2

Introduction

These exercises provide hands-on experience using IBM Tivoli Monitoring, the Tivoli Enterprise Portal, and the NetView Enterprise Management Agent. After completing this unit, you should be able to perform the following tasks:

- Open the Tivoli Enterprise Portal client.
- Describe the components of the Tivoli Enterprise Portal application window.
- Switch Navigator views.
- Navigate workspaces by using Navigator items and links.
- Open the NetView Enterprise Management Agent workspaces.

During these exercises, you navigate many of the options that are available in the Tivoli Enterprise Portal application window. Whenever you are unsure of what you see, access the online help and try to discover the answer before you ask the instructor.

For these student exercises, you primarily use the Windows desktop to access Tivoli Enterprise Portal. However, you can perform all exercises in browser mode as well.

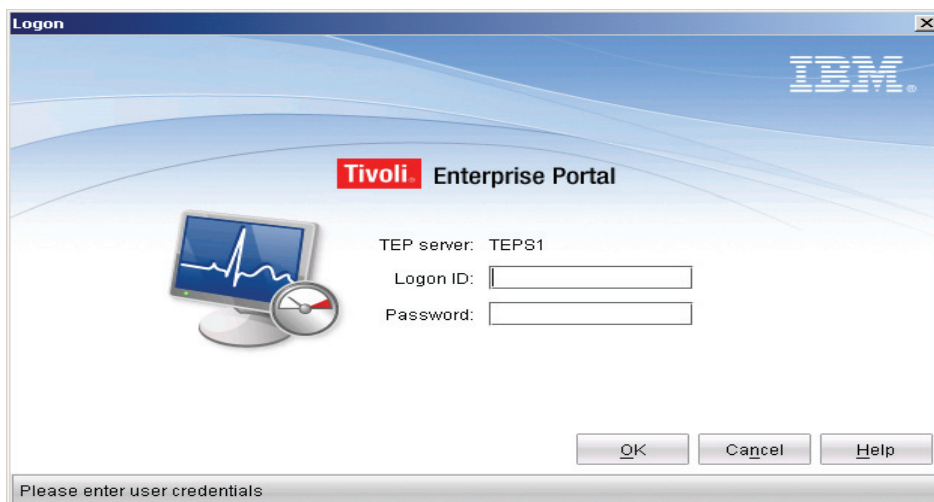
Exercise 2-1: Launching the Tivoli Enterprise Portal client in desktop mode

Browser mode is the most common access method for Tivoli Enterprise Portal. It provides the ability to use workspaces as web addresses. It can simplify maintenance of clients. It detects a missing client or a server release change, and installs or updates clients at connection time. Perform the following steps:

1. Start the Tivoli Enterprise Portal desktop on the VMWare guest desktop.



2. The logon screen prompts for a user ID and password.



3. In your environment, enter **student user** ID and RACF password.

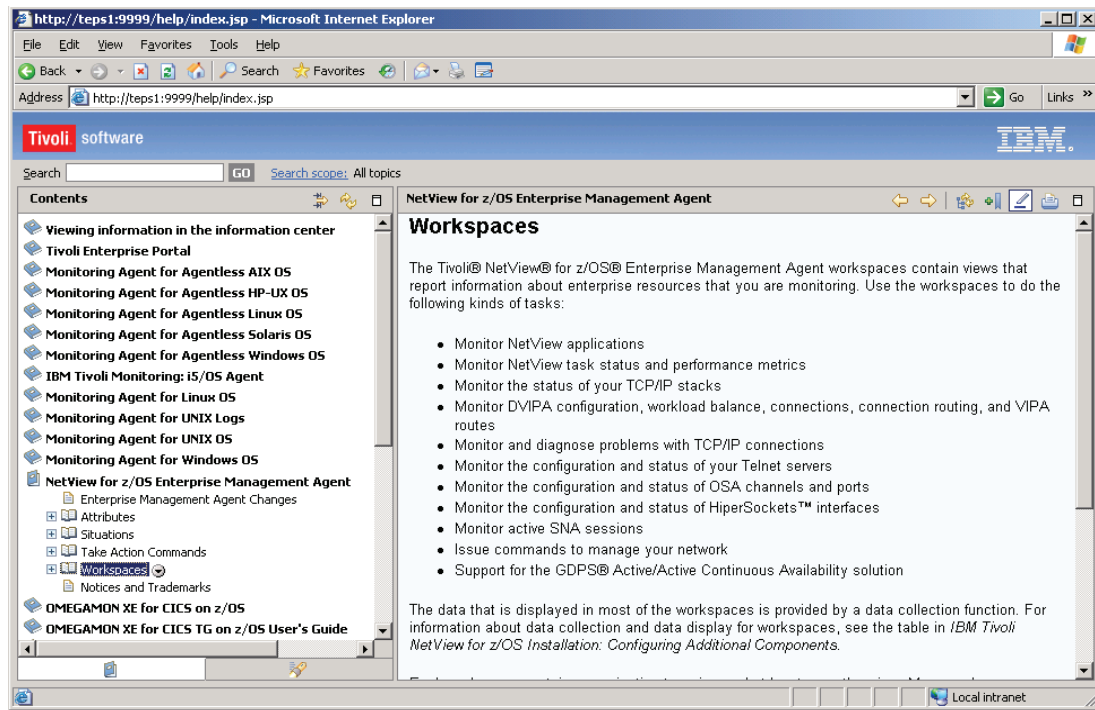
Exercise 2-2: Using the online documentation and help

1. Access and explore the online help window by pressing F1.

When you access the help from the main Tivoli Enterprise Portal window, it opens to a default page. You can also press F1 on specific menus to open the context help that applies to the specific area of interest.

When using new functions in Tivoli Enterprise Portal for NetView, use the online documentation to get started. You can learn all about the advanced options that are available.

2. Access and explore the **IBM Tivoli NetView for z/OS Enterprise Management Agent** area to see more information about the use of the product. For example, click **Workspaces** to see the various types of NetView workspaces.



Note: When accessing help in browser mode, you might want to use the **Help** menu item of the Tivoli Enterprise Portal pane. By pressing F1, you can also access the Microsoft help for Internet Explorer.

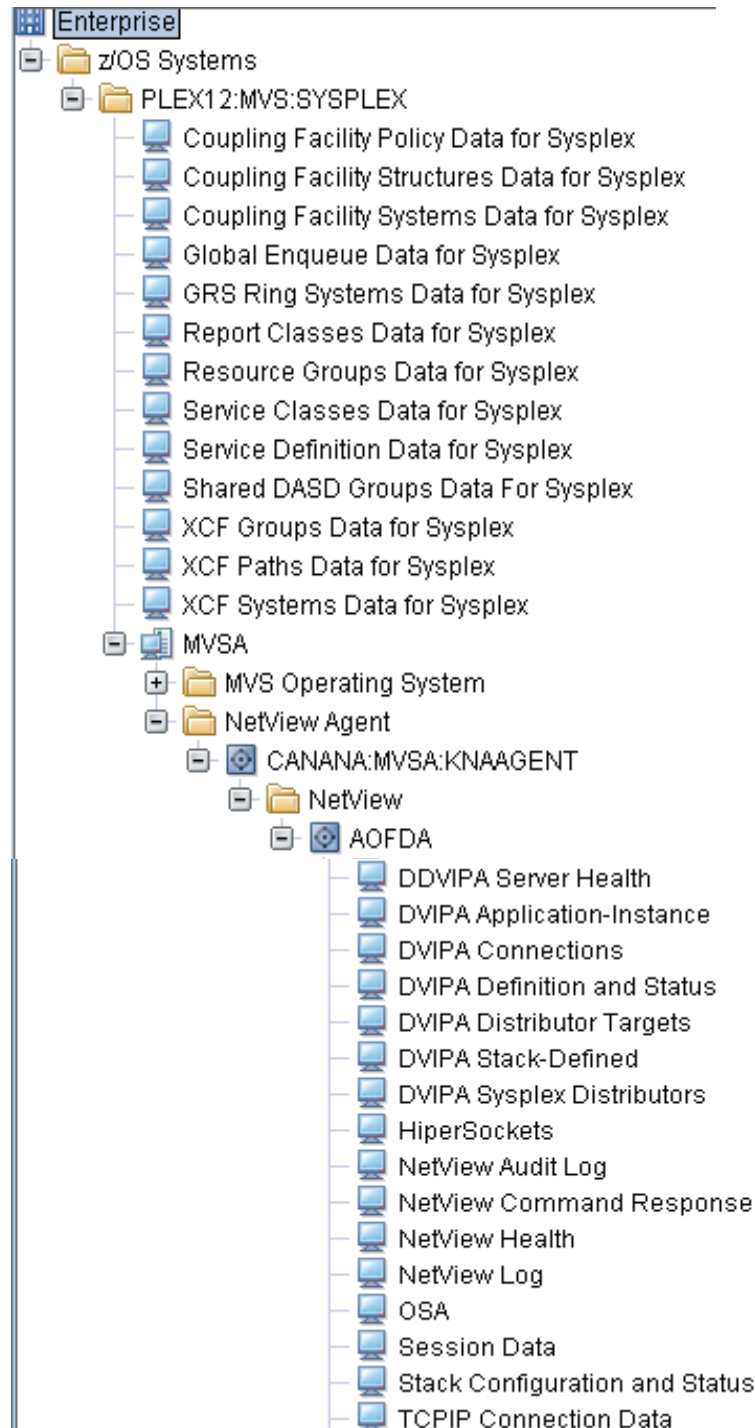
Exercise 2-3: Navigating workspaces

Accessing default workspaces

Each Navigator view contains Navigator items that are associated with workspaces. Use Navigator items to navigate to those workspaces. To view the full structure of a navigator, click the plus sign (+) to expand. Click the minus sign (-) to collapse subtrees.

To access the default workspace associated with a particular Navigator item, click the item.

- ___ 1. Click the navigator physical view to access product-provided workspaces.




- ___ 2. Access the NetView Navigator items by expanding the z/OS Systems tree. Continue to expand at each level. **NetView Agent** is below your host name.

3. Expand the NetView Agent information. Expand the navigator entry, beginning with **CANANA**. Expand the navigator beginning with **NetView**. Finally, expand the Navigator item that references your NetView application domain name.
4. Open a workspace by clicking it. For example, click the **NetView Health** workspace. Each workspace contains views. The NetView Health workspace contains bar chart views for **CPU Utilization >= Critical CPU Util Threshold** and **Storage >= Critical Storage Threshold**. A table view opens for the **NetView Applications Summary**.
















Navigating workspaces by using links

Some product-provided workspaces have table links. From these links, you can navigate to other workspaces that have related or more detailed information. The link icon displays as a yellow part of a chain.

5. Open the NetView Health workspace. On the bottom of this workspace, find a table view called NetView Applications Summary.

NetView Applications Summary														
	Update Time	Domain Name	Role	Sysplex Role	Sysplex Rank	XCF Group List	Total CPU	Total Storage	zOS Image Name	Status	Network ID	RMTCMD IP Address	RMTCMD Port	NetView Version
	11/02/10 02:52:16	AOFDA	NETWORK	MASTER	1	(DSIPLX01)	0	18665	MVSA	ACTIVE	USIBMES	10.31.186.57	4022	V5R4

6. Click the link to the left of the first row. This action opens a list of tasks and links that are associated with every entry.

NetView Tasks Summary													
	Collection Time	Domain Name	Task Name	Status	CPU Utilization	Storage	Message Queue Count	Output Message Rate	Input Message Rate	I/O Rate	Critical CPU Util Threshold	Critical Storage Threshold	
	11/02/10 03:06:13	AOFDA	MAINTASK	ACTIVE	0	3078	0	0	0	0	95	0	
	11/02/10 03:06:13	AOFDA	AOFDAPPT	ACTIVE	0	603	0	0	0	0	95	0	
	11/02/10 03:06:13	AOFDA	DSIMONIT	ACTIVE	0	4	0	0	0	0	95	0	
	11/02/10 03:06:13	AOFDA	DSITIMMT	ACTIVE	0	24	0	0	0	0	95	0	
	11/02/10 03:06:13	AOFDA	DSIDCBMT	ACTIVE	0	3	0	0	0	0	95	0	
	11/02/10 03:06:13	AOFDA	DSIHLLMT	ACTIVE	0	4	0	0	0	0	95	0	
	11/02/10 03:06:13	AOFDA	DSISTMMT	ACTIVE	0	3	0	0	0	0	95	0	
	11/02/10 03:06:13	AOFDA	DSILOGMT	ACTIVE	0	4	0	0	0	0	95	0	
	11/02/10 03:06:13	AOFDA	DSILOG	ACTIVE	0	65	0	0	0	0	95	0	
	11/02/10 03:06:13	AOFDA	DSITRACE	INACTIVE	0	0	0	0	0	0	95	0	
	11/02/10 03:06:13	AOFDA	DSIELTSK	INACTIVE	0	0	0	0	0	0	95	0	
	11/02/10 03:06:13	AOFDA	DSICORSV	INACTIVE	0	0	0	0	0	0	95	0	
	11/02/10 03:06:13	AOFDA	INSTZ24	ACTIVE	0	312	0	0	0	0	95	0	
	11/02/10 03:06:13	AOFDA	AUTIPMGT	ACTIVE	0	468	0	0	0	0	95	0	
	11/02/10 03:06:13	AOFDA	AUTOAON	ACTIVE	0	773	0	0	0	0	95	0	


7. Hover your mouse pointer over the *link to* symbol in one row of the table. Do not click the link yet.

The name of the link representing the workspace name is displayed.




8. Click the link to access the workspace that it points to. This workspace has views for the specific task: CPU Utilization, Storage, Message Queue Count, Message Rate, and I/O Rate.

Accessing other workspaces that a Navigator item associates with

9. Click the TCPIP Connection Data workspace.
10. Right-click the **TCPIP Connection Data** item in the Navigator.

NetView Applications Summary														
	Update Time	Domain Name	Role	Sysplex Role	Sysplex Rank	XCF Group List	Total CPU	Total Storage	zOS Image Name	Status	Network ID	RMTCMD IP Address	RMTCMD Port	NetV Versi
	11/02/10 02:52:16	AOFDA	NETWORK	MASTER	1	(DSIPLX01)	0	18665	MVSA	ACTIVE	USIBMES	10.31.186.57	4022	V5R4

11. Click the **Workspace** option. A list menu displays more workspaces.
12. Click the **Inactive TCP/IP Connection Data** workspace. Each of the items in the table represent an inactive connection. If your table is empty, go to item 14.
13. To see information for a single entry in that table, click the **Link To** icon at the beginning of the row. A filter window opens. Accept the defaults. The window filters the information for this local IP address, local port, remote IP address, and remote port.
14. Access the default workspace for **TCP Connection Data**. A table and rows within the table have a link associated with them.
15. From the **TCPIP Connection Data Summary**, follow the links to more detailed workspace views by right clicking the **Link To** symbol.

TCPIP Connection Data Summary											
	Collection Time	TCPIP Job Name	Local IP Address	Local Port	Remote IP Address	Remote Port	Connection Start Time	Last Activity Timestamp	Resource Name	Connection ID	Total Bytes Received
	11/02/10 03:30:39	TCPIP	10.31.186.57	1918	10.4.127.184	49370	11/02/10 02:48:43	11/02/10 03:30:12	CANADSST	0X00000A03	48734
	11/02/10 03:30:39	TCPIP	10.31.186.57	1918	10.31.186.57	1038	11/01/10 07:15:35	11/02/10 03:29:24	CANADSST	0X00000928	198013
	11/02/10 03:30:39	TCPIP	10.31.186.57	1038	10.31.186.57	1918	11/01/10 07:15:35	11/02/10 03:29:24	CANANA	0X00000927	84716

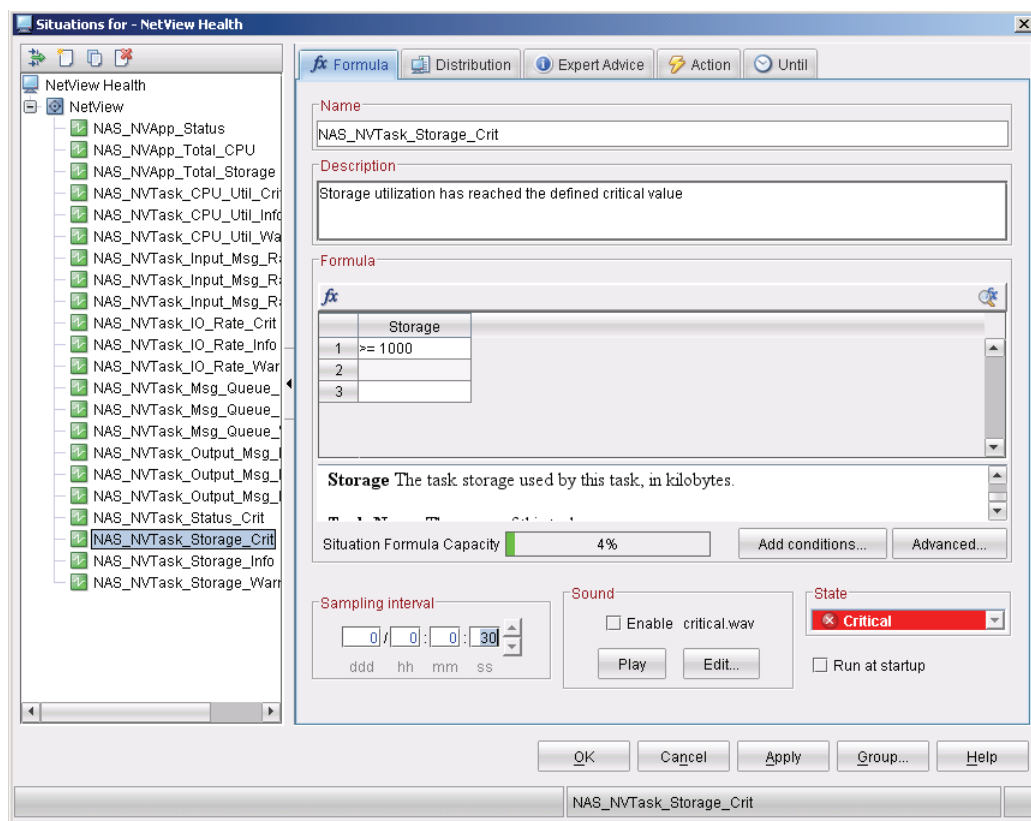
On our system, we have installed Application Support for all the other OMEGAMON XE products. Currently the agents are not installed. If the OMEGAMON Extended Edition (XE) products are installed and properly configured, links to those products and their workspaces are available if the agents are running.

Exercise 2-4: Working with NetView Health

Changing Situation values

In this exercise, modify a threshold to cause an event to occur. Set an artificially low value to generate the event. Perform the following steps:

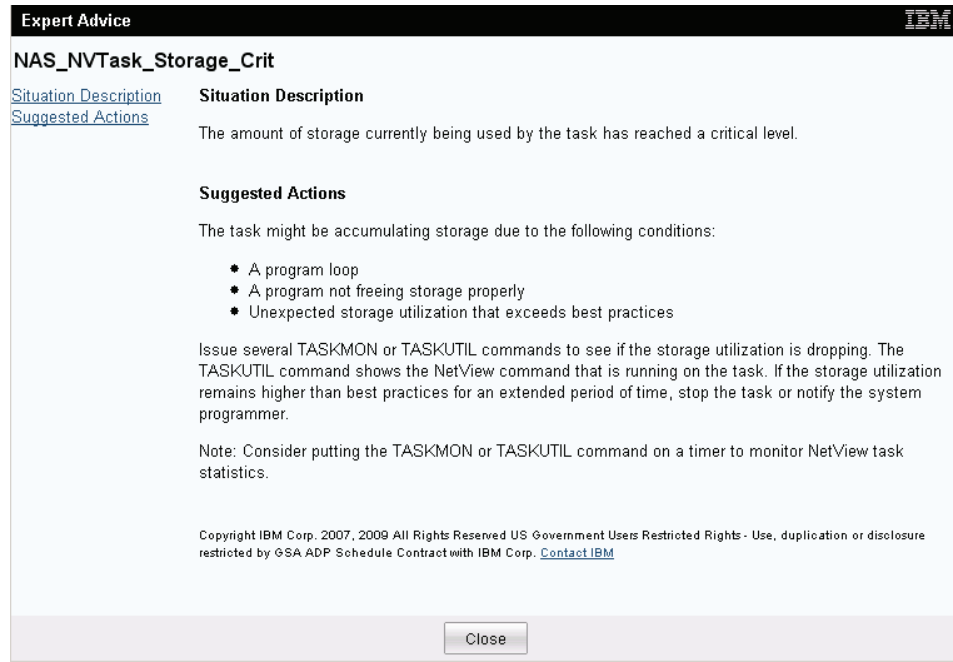
- ___ 1. From the navigator, click **NetView Health** to see the current health. View the numbers from the **Storage >= Critical Storage Threshold** view. Consider a number that selects only a few tasks if it is used as a high threshold value. For example, 1000 kilobytes can do this.
- ___ 2. From the navigator, right-click **NetView Health**.
- ___ 3. From the list menu, click **Situations**.
- ___ 4. In the list of situations, click **NAS_NVTask_Storage_Crit**. Information for this situation is displayed.
- ___ 5. Click the **Formula** tab to see the formula that is used for determining a critical alert. Click the number in the box to edit it. To ensure that a situation event is generated, change the number to something very low, for example, to **>= 1000**.



- ___ 6. Change the sampling interval default (15 minutes) to 30 seconds.
- ___ 7. Enable the sound for this warning. If you want to hear this sound, click **Play**.
- ___ 8. Click **Apply** to save the changes, and click **OK** to leave the Situation editor.
- ___ 9. Check that the situation is running. If a situation is running before you change it, it restarts with the new values. Right-click **AOFDA** in the Navigation tree and click **Manage Situations**. Find the **NAS_NVTask_Storage_Crit** situation in the list and start it by marking it and clicking the green arrow icon at the top left corner.

Using Expert Advice

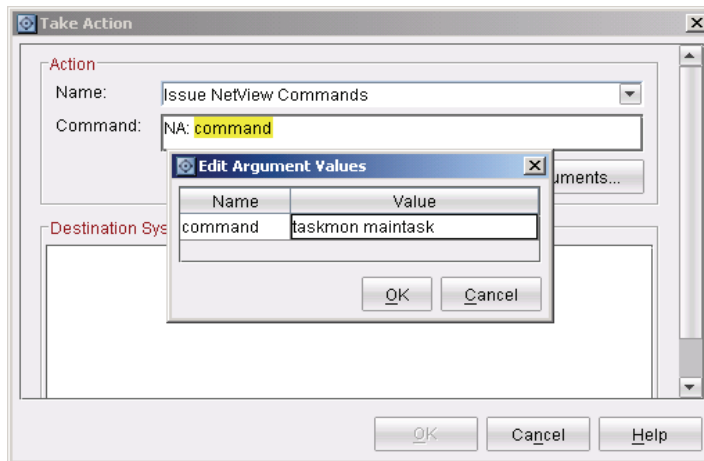
- ___ 10. Around 30 seconds later, the red critical icon displays over many items in the Navigator. The audible alert sounds. Hover the cursor over one of these icons to cause a fly-over box to display.
- ___ 11. In the fly-over box, click the yellow link icon. This action opens a workspace to display the task that exceeded the threshold. The deployable views are Initial Situation Values, Current Situation Values, Command View, and Expert Advice.



If this is a production situation that is being analyzed, the expert advice can lead you to a solution. From the **Command** view, you can issue a **Take Action** command. When the situation resolves, the critical event is no longer displayed in the Situation views.

Issuing a command and viewing the command response

- ___ 12. In this artificial example, try issuing the **TASKMON** or **TASKUTIL** commands to monitor the storage. In the **Command** view, click the **Action Name** list. Click **Issue NetView Commands**.
- ___ 13. In the Edit Argument Values window that opens, perform the following steps:
 - ___ a. For Name, type TASKMON.
 - ___ b. For Value, type MAINTASK, which is the name of the task to monitor.
 - ___ c. Click **OK** to accept the command.
- ___ 14. In the **Command** view, click **Run** to issue the command. In the Action Status window, click **OK**.



- ___ 15. To view the results of the command, in the Navigator, click the **NetView Command Response** workspace. The command and the response are displayed in the **NetView Command Response Summary** view. It might be necessary to change the page number in the upper right part of the view to see the last page.

Exercise 2-5: Refreshing workspaces

Manually refreshing workspaces

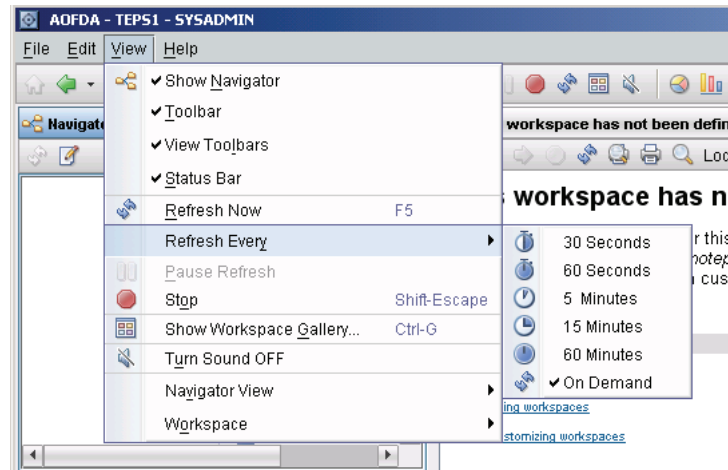
If no automatic refresh interval is specified, you can refresh a workspace by pressing F5. You can also click the refresh icon to view the most current values. Click the icon and look for refreshed data in a view.



Automatically refreshing workspaces

Perform the following steps:

1. To automatically refresh a workspace, click **View > Refresh Every**, and click a time interval from the list.



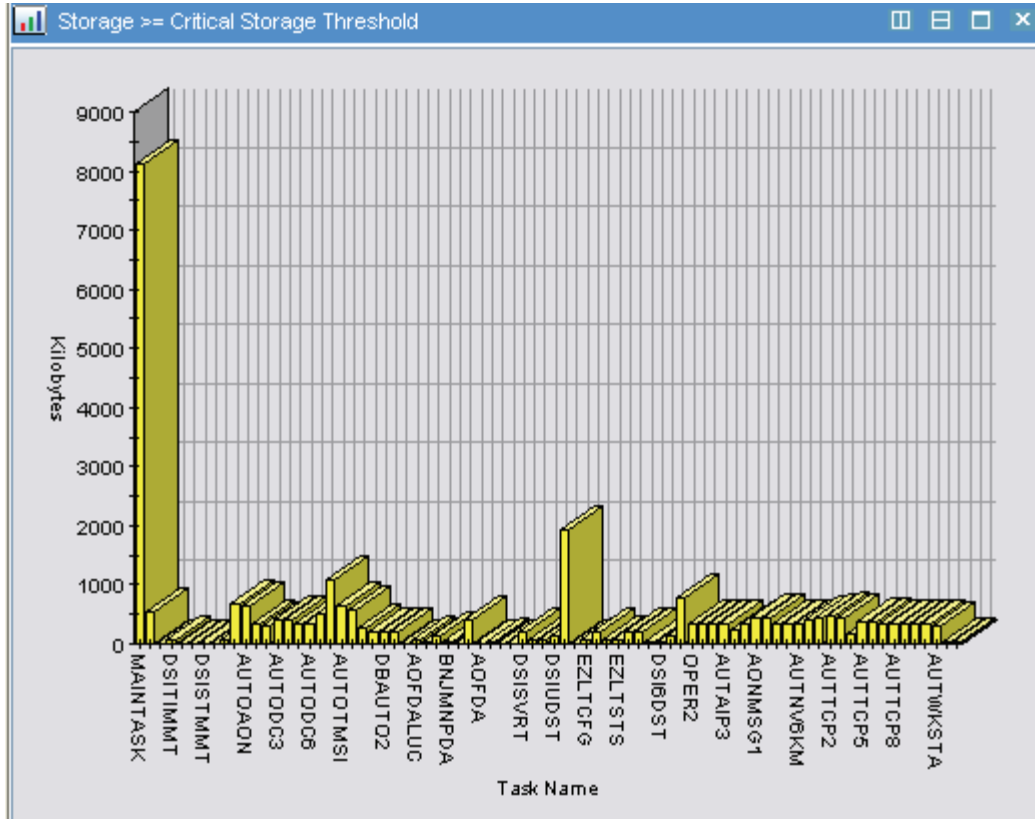
2. Click **30 Seconds**, and monitor the changes. The interval changes apply for the active workspace.
3. To keep changes to your workspace, save the workspace. Give the workspace a meaningful name to easily identify it. Open the workspace again to see that the changes remain.

Exercise 2-6: Using a 3270 view

There might be times when you want to use a 3270 application. You can do that in a workspace view. Perform the following steps:

1. Access the **NetView Health** navigator and the workspace for NetView Tasks.

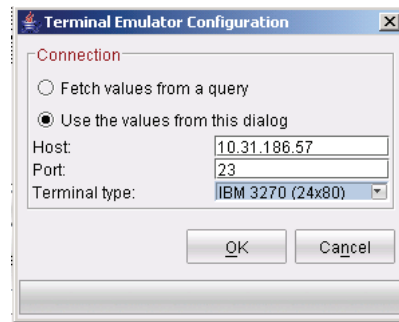
The upper right part of the screen displays a bar chart in the **Storage >= Critical Storage Threshold** view.



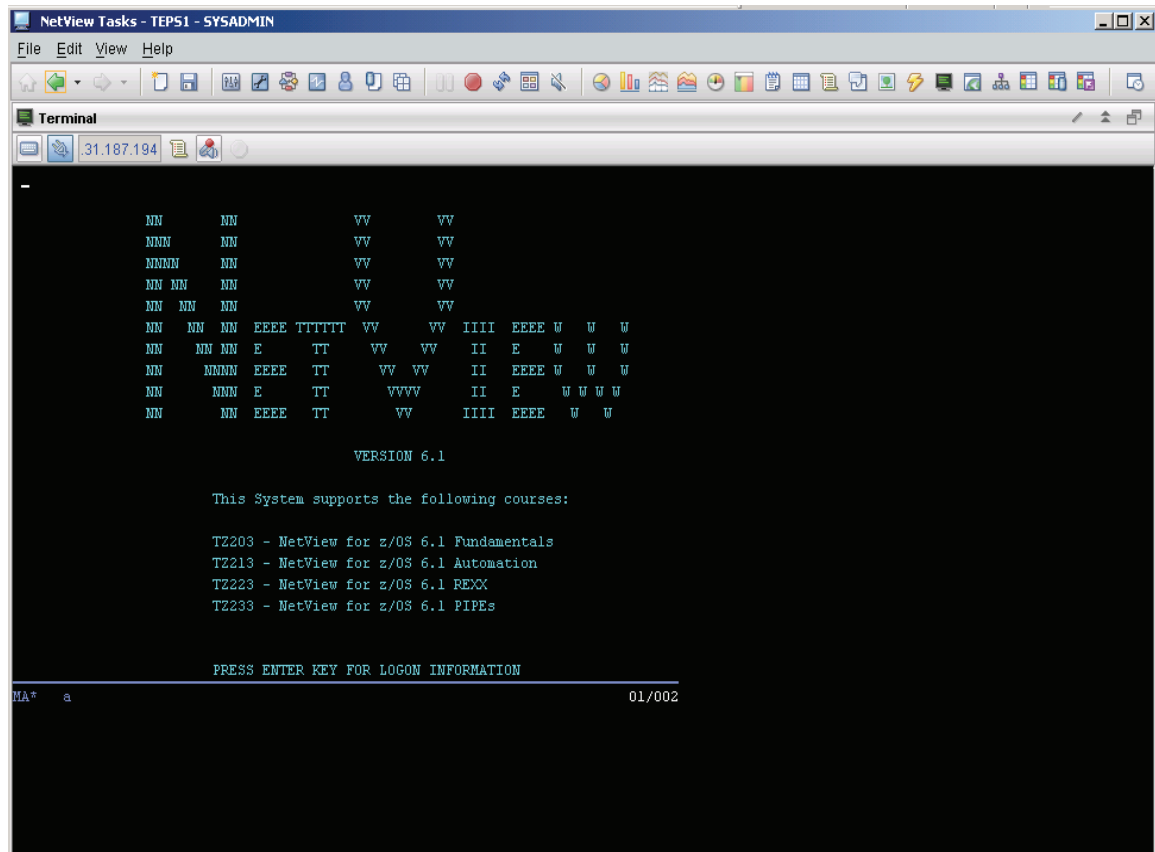
2. At the toolbar, click the terminal icon. The mouse icon changes to the terminal icon. Move your mouse back to the **Storage >= Critical Storage Threshold** view, and click the mouse button within the view.



3. In the Terminal Emulator Configuration window that opens, type the IP address that your instructor provided for your 3270 Personal Communications address.



4. In the **Terminal Type** list, click **IBM 3270 (32 x 80)**, and click **OK**.



It might be necessary to widen the screen height to see the entire 3270 screen.

5. Using the 3270 workstation view, log on to NetView and issue the BROWSE NETLOGA command.



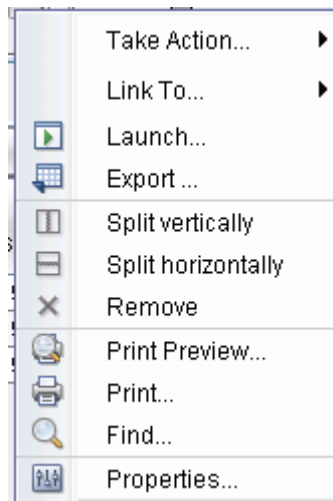
Note: The workspace you have modified is your own private copy. If you want to let others access it, you must enter Workspace Administration Mode. This is though not covered in this course.

Exercise 2-7: Queries

Accessing queries

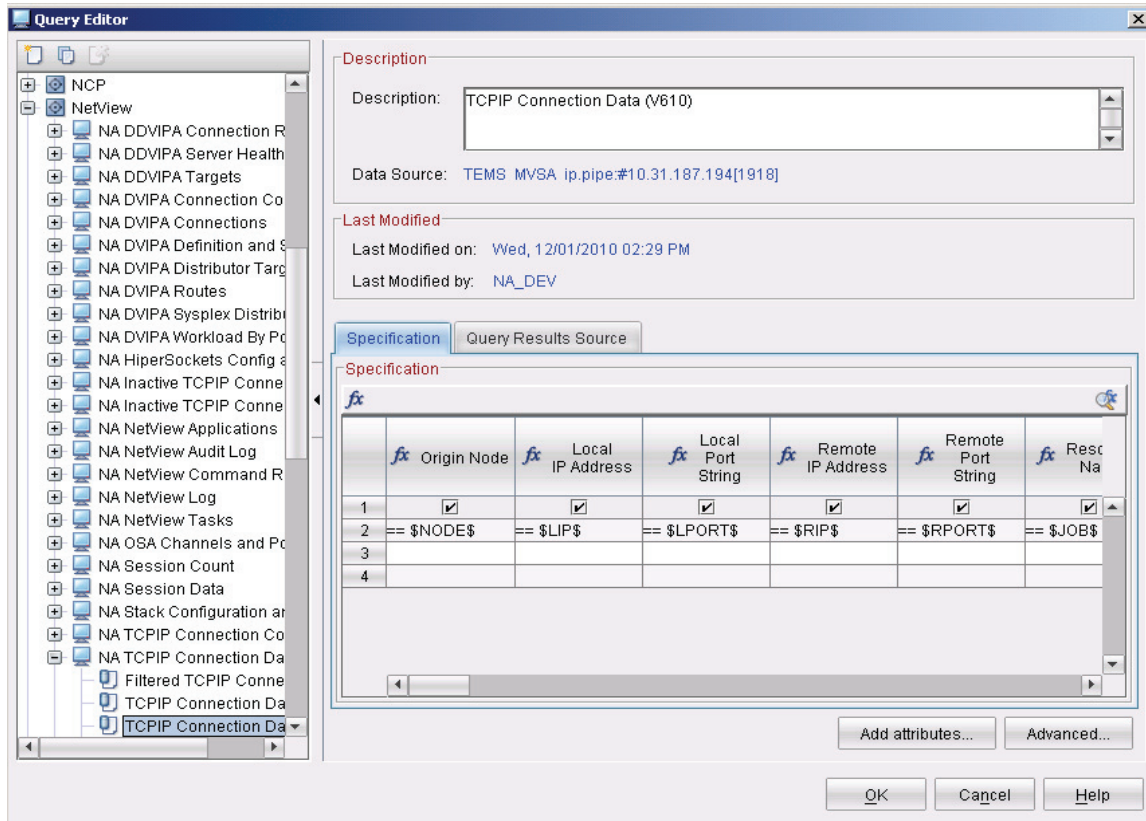
Build a workspace based on queries that are associated with the included views. You might not modify or use queries when you do not work with Tivoli Enterprise Portal as an administrator. Certain data is available for a specific view, such as the example that follows: Perform the following steps:

- ___ 1. Access the **TCPIP Connection Data** workspace.
- ___ 2. From within the **TCPIP Connection Data Summary** view, access the right-click menu.



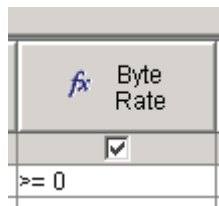
- ___ 3. Click **Properties** to open the **View Properties Editor**. The first tab in properties displays the query that is associated with the view. In this case, the formula shows the query that is to be sent to the agent.

4. Click **Click here to assign a query**. This opens the Query editor.



The Byte Rate heading has a value of **>= 1024**. When the byte rate is greater than or equal to 1024, that table entry is displayed.

5. To limit the output that displays, modify this byte rate selection query to a higher number, such as **>= 4096**. If none are selected, change it to another number until the table shows more rows.



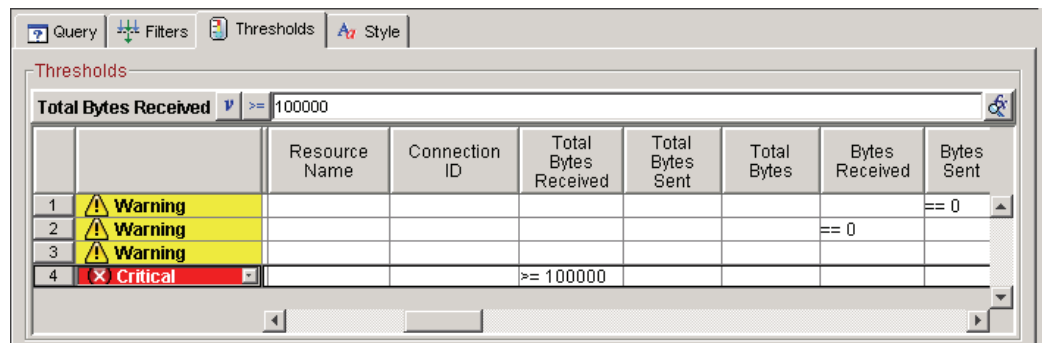
Click **OK** to save the query selection. From the Properties window, you can test by clicking **Test**. Click **Apply**. You should see fewer lines of data that match this selection criteria.

6. If you do not see the **Byte Rate** column in the display, it may be getting filtered. Click the **Filter** tab, scroll to the right, and ensure **Byte Rate** is selected. Click **Test**. Click **Apply** to keep the filter criteria.

Using thresholds

You can use a threshold. Some of the tables that were shown in the previous workspaces have color-coded fields. Those colors result from thresholds that have been set for a view.

- ___ 7. Click the **TCPIP Connection Data** workspace. In the **TCPIP Connection Data Summary** view, right-click from within the table and select **Properties**.
- ___ 8. Click the **Threshold** tab. On the line displaying **Critical**, scroll to the right and change the **Total bytes received** to greater than or equal to 100000.



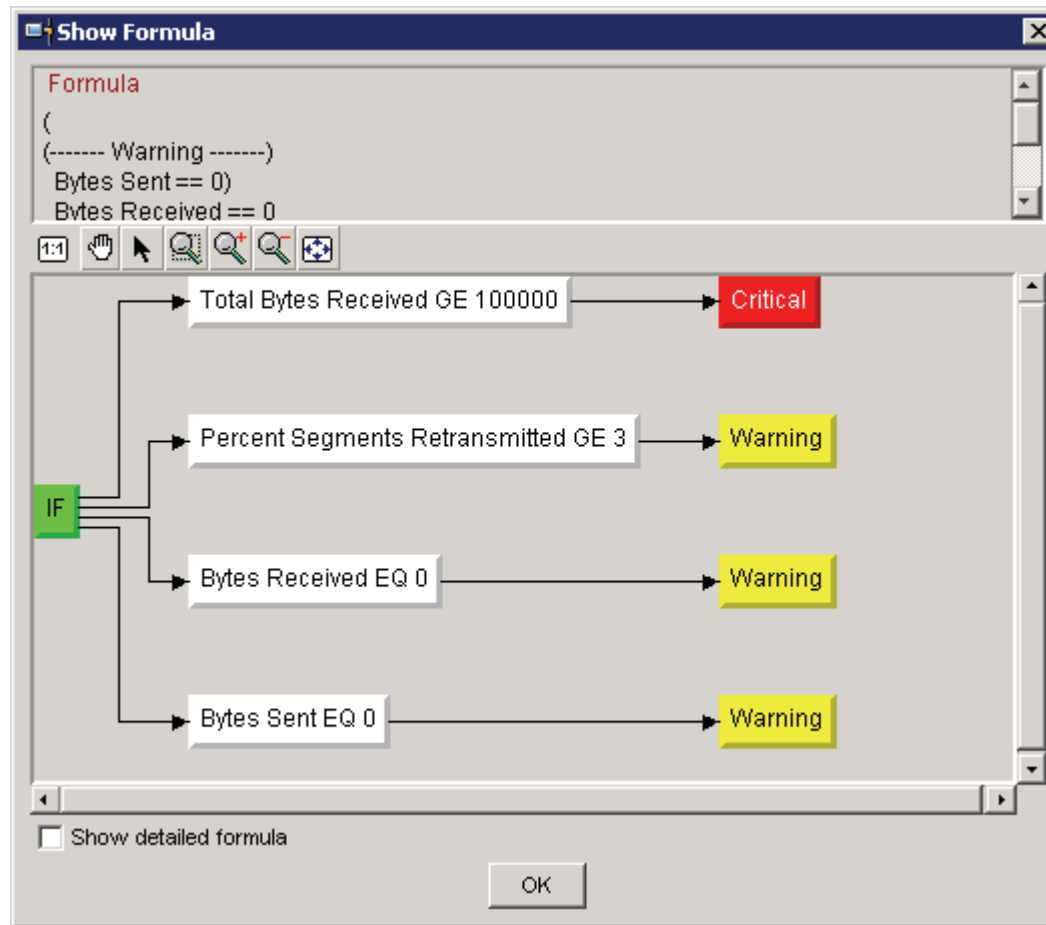
		Resource Name	Connection ID	Total Bytes Received	Total Bytes Sent	Total Bytes	Bytes Received	Bytes Sent
1	Warning						== 0	
2	Warning						== 0	
3	Warning							
4	Critical			>= 100000				

- ___ 9. Click the **Test** button and look for anything that changes to a red color. If nothing is red in the top table, lower the number and try again until you see red in the table. Click **Apply** and click **OK** to exit. In the table, those entries that match this threshold setting have red Total Bytes Received cells.
- ___ 10. You can see the Threshold selections as a formula. Perform the following steps:
 - ___ a. Enter the **Properties** again for the **TCPIP Connection Data Summary** view.
 - ___ b. Click the **Threshold** tab.
 - ___ c. Click the icon to view the threshold conditions in a more readable format. (The icon is under the **Thresholds** toward the right side.)





Note: Some of the threshold rules might be complex.



With what you have learned, try changing thresholds on your own.

- ___ 11. In the **DVIPA Connections**, change the threshold indicator for Total Bytes Received to greater than or equal to 4096 as a critical threshold.

Exercise 2-8: Other Tivoli Enterprise Portal-related functions (optional)

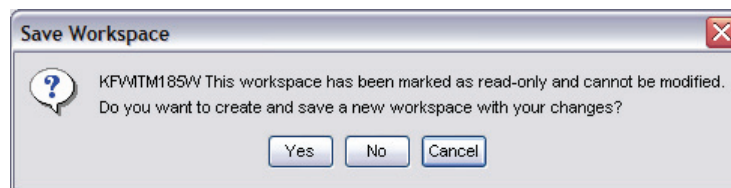
In this exercise, you learn about making minor modifications to workspaces, and about data collection.

Saving workspaces

When you make changes to a workspace, you need to save them to keep the changes. You save a workspace by clicking the **Save** button or pressing Ctrl+S on the keyboard.



Click a workspace and click the **Save** button. A confirmation window opens.



You can save your own copy and prevent original workspaces from being replaced. Workspaces depend on the user ID. Therefore, modifications apply to only your user ID. Only an administrator who creates a workspace as another user and publishes it to others can see the same workspace.

Accessing workspaces as web addresses



Note: This feature is available only when using the browser mode to access the Tivoli Enterprise Portal client.

When using browser mode, each workspace is accessible with a specific web address. You can create entries in your Favorites that provide easy navigation to workspaces of interest to you. Perform the following steps:

- ___ 1. Open the **DVIPA Distributor Targets** workspace. Save it to your favorites by clicking **Favorites > Add to Favorites**.
- ___ 2. Type a name that is meaningful to you, and add it to a folder where you can locate it again.

- ___ 3. Access a different workspace.
- ___ 4. Click the newly created entry in your **Favorites**. This action returns you to the **DVIPA Distributor Targets** workspace.
- ___ 5. This feature also works when you are not logged on to a client and selecting it with a browser. You receive a prompt for an ID and password before the workspace displays. To test, log off from the client and click a saved workspace from your Internet Explorer Favorites.

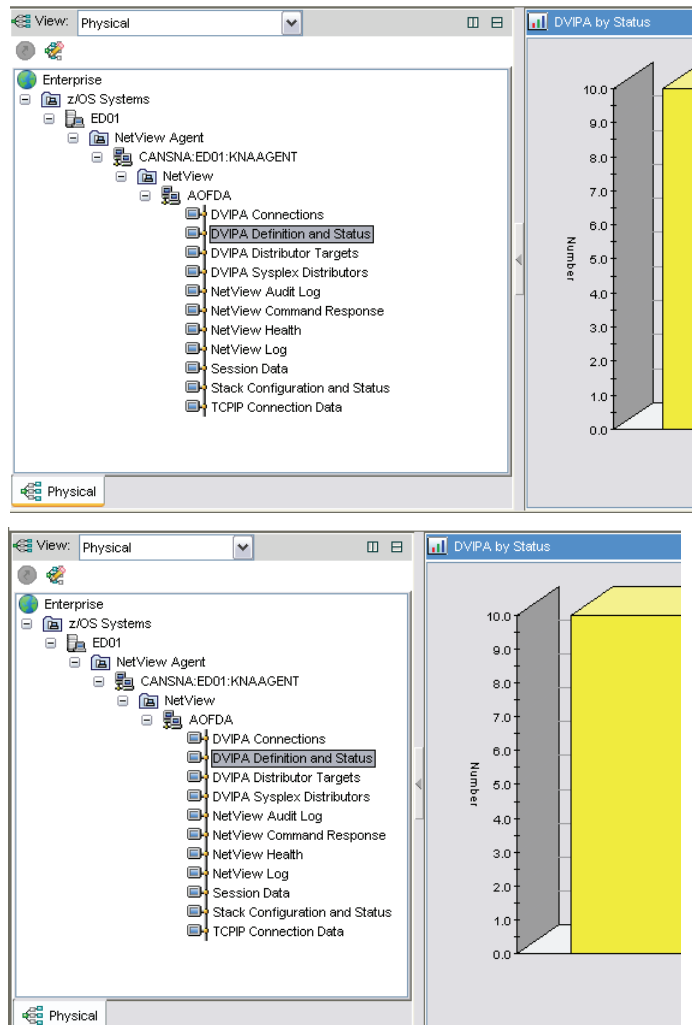
Modifying view panes

Not all aspects of modifying views are covered in this unit. However, there are some modifications that users can do.

Resizing views

One of most common scenarios that you encounter is a different screen resolution on your desktop. Although most workspaces have been created with an industry standard screen size in mind, it might be beneficial for you to adjust the view sizes. This also applies if you do not want to view Tivoli Enterprise Portal in full-screen mode, or if you use the browser mode.

Hover your mouse pointer on the border of a view and drag it in a direction that the arrow points. In the following example, the bar that separates the Navigator from the view to the right has been dragged to the left. This action reduces the size of the Navigator.



After you adjust view sizes, you must save your workspace to keep your changes. Press the **Save** button.

Modifying table views

When working with table views, you can modify the column width, sort order, and sequence of columns.

Resizing columns

Access a workspace that contains a table, for example, the **TCPIP Connection Data**. Hover your mouse pointer on the border between two column headings. Click and drag your mouse to resize your table row. In the following example, the **Collection Time** column has narrowed.

TCPIP Connection Data Summary

	Collection Time	TCPIP Job Name	Local IP Address	
	09/20/07 13:40:25	TCPIP	10.44.15.204	
	09/20/07 13:40:25	TCPIP	10.44.15.204	
	09/20/07 13:40:25	TCPIP	10.44.15.204	




TCPIP Connection Data Summary





	Collection Time	TCPIP Job Name	Local IP Address	Local P
	09/20/07 13:40...	TCPIP	10.44.15.204	19
	09/20/07 13:40...	TCPIP	10.44.15.204	19
	09/20/07 13:40...	TCPIP	10.44.15.204	10





Sorting columns

By pressing any column heading, the sort order switches among not sorted, ascending sort order, and descending sort order. From the **TCPIP Connection Data** view, click the **Remote IP Address** column heading several times and view the changes.

The first example shows sorted order, the second ascending, and the third descending.




TCPIP Connection Data Summary						
	Collection Time	TCPIP Job Name	Local IP Address	Local Port	Remote IP Address	Remote Port
	09/20/07 13:40:25	TCPIP	10.44.15.204	1918	10.44.15.204	1076
	09/20/07 13:40:25	TCPIP	10.44.15.204	1918	9.27.132.48	1323
	09/20/07 13:40:25	TCPIP	10.44.15.204	1076	10.44.15.204	1918




TCPIP Connection Data Summary						
	Collection Time	TCPIP Job Name	Local IP Address	Local Port	 Remote IP Address	Remote Port
	09/20/07 13:40:25	TCPIP	10.44.15.204	1918	10.44.15.204	1076
	09/20/07 13:40:25	TCPIP	10.44.15.204	1076	10.44.15.204	1918
	09/20/07 13:40:25	TCPIP	10.44.15.204	1918	9.27.132.48	1323

TCPIP Connection Data Summary						
	Collection Time	TCPIP Job Name	Local IP Address	Local Port	 Remote IP Address	Remote Port
	09/20/07 13:40:25	TCPIP	10.44.15.204	1918	9.27.132.48	1323
	09/20/07 13:40:25	TCPIP	10.44.15.204	1918	10.44.15.204	1076
	09/20/07 13:40:25	TCPIP	10.44.15.204	1076	10.44.15.204	1918

Changing column sequence

In the same workspace view, click the column heading labeled **Remote IP Address** and hold down the left mouse button. Drag that column to the front of the **Local IP Address** column and release the button.

TCPIP Connection Data Summary						
	Collection Time	TCPIP Job Name	Local IP Address	Local Port	Remote IP Address	Remote Port
	09/20/07 14:57:08	TCPIP	10.44.15.204	1918	10.44.15.204	1085
	09/20/07 14:57:08	TCPIP	10.44.15.204	1085	10.44.15.204	1918
	09/20/07 14:57:08	TCPIP	10.44.15.204	1918	9.27.132.48	1118

TCPIP Connection Data Summary						
	Collection Time	TCPIP Job Name	Remote IP Address	Local IP Address	Local Port	Remote Port
	09/20/07 14:57:08	TCPIP	10.44.15.204	10.44.15.204	1918	1085
	09/20/07 14:57:08	TCPIP	10.44.15.204	10.44.15.204	1085	1918
	09/20/07 14:57:08	TCPIP	9.27.132.48	10.44.15.204	1918	1118



IBM Tivoli NetView for z/OS 6.1: Automation Techniques

Student Exercises

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

|||||

No student exercises are provided for this unit. 1-1

Lab exercise overview	2-1
Data sets used in This lab	2-1
Exercise 2-1: Message Revision Table (MRT)	2-1
Lab exercise objectives	2-1
Lab exercise instructions	2-2
Lab exercise overview	2-2
Payroll start messages	2-3
Payroll stop messages	2-3
Payroll cancel messages	2-4
Additional messages issued by payroll	2-4
Connecting to TSO	2-5
Connecting to NetView (AOFDA)	2-5
Testing your MRT	2-6
Answers to Lab 2 Exercise 1 questions	2-9
Example MRT (VAPMRT1)	2-10
Exercise 2-2: ASSIGN Command	2-11
Lab exercise objectives	2-11
Lab exercise instructions	2-11
Answers to Lab 2 Exercise 2 questions	2-15
Exercise 2-3: NetView and z/OS consoles	2-16
Lab exercise objectives	2-16
Lab exercise instructions	2-16
Answers to Lab 2 Exercise 3 questions	2-19

Lab exercise overview	3-1
Exercise 3-2: Routing commands to other tasks and domains	3-1
Lab exercise objectives	3-1
Lab exercise instructions	3-1
Answers to Lab 3 Exercise 2 questions	3-3
Exercise 3-3: Timer commands	3-4
Lab exercise objectives	3-4
Lab exercise instructions	3-4
Answers to Lab 3 Exercise 3 questions	3-8
Exercise 3-4: Command Revision Table (CRT)	3-9
Lab exercise objectives	3-9
Lab exercise instructions	3-9
Answers to Lab 3 Exercise 4 questions	3-11

Lab exercise overview	4-1
Exercise 4-1: Managing the NetView automation tables	4-1

Lab exercise objectives	4-1
Lab exercise instructions	4-1
Answers to Lab 4 Exercise 1 questions	4-4

Student exercises for Unit 5

Lab exercise overview	5-1
Data sets used in this lab	5-1
Exercise 5-1: Coding automation table statements	5-1
Lab exercise objectives	5-1
Lab exercise instructions	5-1
Lab exercise instructions	5-2
Lab exercise instructions	5-4
Answers to Lab 5 Exercise 1 Questions	5-6

Student exercises for Unit 1

No student exercises are provided for this unit.

Student exercises for Unit 2

.....

Lab exercise overview

The z/OS console is a major source of messages for automation. The messages can be modified or suppressed before they route to NetView for automation. The messages can also be suppressed from automation. In this exercise, you create message revision table (MRT) statements for processing messages that route to the MRT. During the lab exercise, you use the two following PCOMM sessions:

- NetView
- TSO

Data sets used in This lab

NV390.WORKSHOP.DSIPARM _____

Exercise 2-1: Message Revision Table (MRT)

Lab exercise objectives

This lab exercise uses the NetView Message Revision Table (MRT) to suppress and modify messages that are to route to the z/OS system console. At the end of the lab, you should be able to perform the following tasks:

- Define MRT statements to suppress messages.
- Define MRT statements to modify messages.
- Activate an MRT.

Lab exercise instructions

Use two PCOMM 3270 emulator sessions as follows:

- A TSO session for creating a Message Revision Table (MRT) member in DSIPARM. If the instructions refer to a panel ID, you can find it in the upper left corner of the panel
- A NetView session for activating the MRT and issuing commands that generate the messages for the MRT. Use the Canzlog data space to verify messages on the system console. If you want to access the system console, ask your instructor.



Note: Messages that are suppressed by MRT will show up in canzlog, by selecting the message and pressing Enter you see if message has been suppressed:

```
CNMK CZMD OUTPUT FOR $HASP395 Time: 07/13/11 05:27:44.523
CzID: 76807 00012C07x      AutoTime: 5 msec      DomTime: none
JobName: PAYROLL          DestConsole:          AutoToken:
Tags: MVS
Flags: MRT, Suppr, Auth
CHkey: PAYROLL            SystemID: MVSA          JobID: STC00578
SmsgID: 52001F4Fx         ASID: 003Dx          DomToken:
000000000x
AStype: S                 AuthUser: SYSPROG        AuthGroup: SYS1
Mtype: E (C5x)

DescCodes: 0400 (6)
RouteCodes: 40000000000000000000000000000000
              (2)

$HASP395 PAYROLL ENDED
```

Lab exercise overview

Use the TSO session to edit VAPMRT1 and create MRT statements for suppressing and revising (modifying) messages. For this lab exercise, you are the system programmer who is responsible for the *payroll* application. When the payroll application starts and stops, several messages are generated. Your job is to suppress some of the messages pertaining to the *payroll* application and revise other messages. You create an MRT to accomplish those tasks.

First, you need to know the messages that are issued for each action and also a description of how to process each message when it runs through the MRT. Some of the messages route to the system log (syslog) only. You code MRT statements to process the syslog messages and messages that go the system console.

Payroll start messages

When the payroll application starts, the following messages are displayed at the z/OS system console:

```
S RUNVAPL,JOBNAME=PAYROLL,NAME=VAPL20
$HASP100 PAYROLL ON STCINRDR
IEF695I START RUNVAPL WITH JOBNAME PAYROLL IS ASSIGNED TO USER
SYSPROG
, GROUP SYS1
$HASP373 PAYROLL STARTED
IEF403I PAYROLL - STARTED - TIME=02.11.00
@0023 VAPL20000A REPLY WARM OR COLD
R 23 SUPPRESSED
IEE600I REPLY TO 0023 IS;SUPPRESSED
+VAPL20010I VAPL20 WARM START COMPLETE
@0024 VAPL20999A REPLY END TO STOP
```

The VAPL29999A message indicates the payroll application is active.

During the lab exercise, you code MRT statements to accomplish the following tasks:

- Suppress all \$HASP100, and \$HASP373 messages.
- Do nothing to any IEF695I message.
- Revise IEF403I to send it to NetView for automation only.

IEF403I is required by SA z/OS. You need to pass it to NetView.

Payroll stop messages

When the payroll application stops by responding to its outstanding WTOR, the following messages are displayed at the z/OS system console:

```
R 24 SUPPRESSED
IEE600I REPLY TO 0024 IS;SUPPRESSED
+VAPL20020I VAPL20 SHUTDOWN COMPLETE
-
--TIMINGS (MINS.)--
----PAGING COUNTS---
-JOBNAME STEPNAME PROCSTEP RC EXCP CONN TCB SRB CLOCK
SERV PG PAGE SWAP VIO SWAPS
-PAYROLL STEP1 3408 6 0 551177 .00 .4
637 0 0 0 0 0
IEF404I PAYROLL - ENDED - TIME=02.11.26
-PAYROLL ENDED. NAME- TOTAL TCB CPU TIME= .00
TOTAL ELAPSED TIME= .4
$HASP395 PAYROLL ENDED
IEA989I SLIP TRAP ID=X33E MATCHED. JOBNAME=*UNAVAIL, ASID=0040.
$HASP250 PAYROLL PURGED -- (JOB KEY WAS C6DA9EB2)
```

During the lab exercise, code MRT statements to perform the following tasks:

- Suppress all \$HASP395 messages.
- Revise IEF404I to send it to NetView for automation only.
IEF404I is required by SA z/OS. You need to pass it to NetView.
- Delete all IEA989I messages.
- Delete all IEE600I messages.
- Revise all \$HASP250 messages so that they show as red and blinking

Payroll cancel messages

When the payroll application is canceled, the following messages appear at the z/OS system console:

```
C PAYROLL
IEA989I SLIP TRAP ID=X222 MATCHED.  JOBNAME=PAYROLL , ASID=0040.
IEE400I THESE MESSAGES CANCELLED - 0029.
IEF450I PAYROLL PAYROLL - ABEND=S222 U0000 REASON=00000000 550
      TIME=02.27.54
-
      --TIMINGS (MINS.)--
      ----PAGING COUNTS----
-JOBNAME  STEPNAME  PROCSTEP    RC    EXCP    CONN    TCB    SRB    CLOCK
SERV  PG  PAGE  SWAP    VIO  SWAPS
-PAYROLL          STEP1    *S222      6      0 551177    .00    .3
588    0    0    0      0    0
-PAYROLL ENDED.  NAME-          TOTAL TCB CPU TIME=    .00
TOTAL ELAPSED TIME=    .3
$HASP395 PAYROLL ENDED
IEE301I PAYROLL          CANCEL COMMAND ACCEPTED
IEA989I SLIP TRAP ID=X33E MATCHED.  JOBNAME=*UNAVAIL, ASID=0040.
```

During the lab exercise, code MRT statements to perform the following tasks:

- Suppress all IEE400I and \$HASP395 messages.
- Revise any IEF450I message to be displayed in yellow if issued for the payroll application. Otherwise, change the color to white for all other applications.
- Delete all IEE301I messages.

Additional messages issued by payroll

The payroll application issues messages that have a prefix of VAPL29. Examples are as follows:

- VAPL29999A indicates that the payroll application is active.
- VAPL29020I indicates that the payroll application stopped.

During the lab exercise, code MRT statements to perform the following tasks:

- Use an UPON(ALWAYS) condition to display all payroll messages in reverse video blue by using one of these methods:
 - Test for a jobname of PAYROLL for a length of eight characters.
 - Test a substring of the message ID for all VAPL29 messages. An example is as follows:

```
(MSGID LEFT 6 = 'VAPL29')
```

Connecting to TSO

Perform the following steps:

- ___ 1. Access your system and log on to TSO, following the instructions that the instructor provides. Edit the VAPMRT1 member in NV390.WORKSHOP.DSIPARM.

Initially, VAPMRT1 has one line in it as follows:

```
* Empty MRT - ready for your input
```

Otherwise, ask your instructor for assistance.

- ___ 2. You can now begin to code your MRT statements in VAPMRT1. Use UPON, SELECT-WHEN, REVISE, NETVONLY, and so on.

Refer to the discussion of the messages that are issued when the payroll application is started, stopped, and canceled for properly coding your MRT statements.



Tip: For additional help, browse the sample MRT supplied by NetView, CNMSMRT1, to help you code your MRT statements.

- ___ 3. Press PF3 to save the MRT when you are ready to test. Do not end your session with TSO, in case you need to make further changes to the VAPMRT1 member.
- ___ 4. Use the Canzlog data space to verify that your MRT logged messages appropriately.

You are now finished creating the MRT statements and need to load the MRT from NetView.

Connecting to NetView (AOFDA)

- ___ 5. Access your system and log on to NetView domain AOFDA following the instructions provided by your instructor.

Testing your MRT

- ___ 6. At this point, you are ready to test the MRT you defined. Type “REVISE STATUS” and verify no MRT loaded. You should see:

```
REVISE STATUS
DSI231I No 'revision MSG table' is active
DSI231I No 'revision CMD table' is active
```

If you do not see the DSI231I message, ask your instructor for assistance.

- ___ 7. Without an active MRT, start the payroll application by issuing the command:

```
PAYROLL START
```

Browse the canzlog and look for the messages pertaining to the start of the payroll application. Also, you may want to look at the system console. Which messages were issued? Note that suppressed messages show up in canzlog, but if you look at the message details it will say suppressed by MRT.



Note: The PAYROLL command is a REXX routine that is written for these lab exercises to simplify the start and stop of the payroll application.

- ___ 8. Stop the payroll application by issuing the following command:

```
PAYROLL STOP
```

Browse the canzlog and look for the messages pertaining to the stop of the payroll application. Also, look at the emulator session for your system console. Which messages were issued? _____

- ___ 9. Restart the payroll application. When it becomes active, cancel it by issuing the following command:

```
PAYROLL CANCEL
```

Browse the canzlog and look for the messages pertaining to the cancelling of the payroll application. Also, look at the emulator session for your system console. Which messages were issued? _____

- ___ 10. Test the MRT that you created. Issue the following command:

```
REVISE MSG, MEMBER=VAPMRT1, TEST
```

The CNM501I message is as follows:

```
CNM501I TEST OF NETVIEW AUTOMATION FILE "VAPMRT1" WAS SUCCESSFUL
```

If you have errors, switch to your TSO session to correct the errors and reissue the REVISE MSG command to test your corrections.



Tip: If you are not sure why you have an error, ask your instructor for assistance.

- ___ 11. When your testing of the MRT yields no errors, activate it by entering the following command:

```
REVISE MSG, MEMBER=VAPMRT1
```

The following BNH608I message is displayed:

```
BNH608I 'LOAD MSG VAPMRT1' REQUEST COMPLETED SUCCESSFULLY
```

- ___ 12. Your MRT is now active. Query the status of the MRT by issuing a REVISE MSG,STATUS command. A message similar to the following one is displayed:

```
CNM012I MSG Revision table VAPMRT1, loaded by TSCCW01, has  
examined
```

```
0 objects since being loaded, 11/09/10 05:03:19
```

This message indicates that the VAPMRT1 Message Revision Table is active but no messages run through the MRT yet.

Test your MRT. Perform all of the following steps before making any changes to your MRT.



Tip: If the expected results do not occur, you can issue the REVISE MSG,REPORT command to determine if the MRT statements were driven and matched.

- ___ 13. Start the payroll application with a PAYROLL START command. Verify that your MRT performed the following tasks:

- Suppressed the \$HASP100 and \$HASP373 messages. These messages should still appear in syslog.
- Took no action with the IEF695I message.
- Revised the IEF403I message to send it to NetView for automation only. This message is to be suppressed and not logged to syslog.

- ___ 14. Stop the payroll application with a PAYROLL STOP command. Verify that your MRT performed the following tasks:
- Suppressed the \$HASP395 messages. This message is to still be displayed in syslog.
 - Revised the IEF404I to send it to NetView for automation only. This message is to be suppressed and not logged to syslog.
 - Deleted the IEA989I message. This message is to be suppressed and not logged to syslog.
 - Deleted the IEE600I message.
 - Revised the \$HASP250 message to be displayed as red and blinking.
- ___ 15. Restart the payroll application. When it becomes active, cancel the payroll application with a PAYROLL CANCEL command. Verify that your MRT performed the following tasks:
- Suppressed the IEE400I and \$HASP395 messages. These messages should still appear in syslog.
 - Revised the IEF450I message to be displayed in yellow.
 - Deleted the IEE301I message. This message is to be suppressed and not logged to syslog.
 - Revised the \$HASP250 message to be displayed as red and blinking.
- ___ 16. If your lab version of VAPMRT1 did not function as expected, perform the following steps:
- ___ a. Switch to your TSO emulator session and make the appropriate changes.
 - ___ b. From your NetView emulator session, issue the REVISE MSG command to test and load the modified VAPMRT1.
 - ___ c. Repeat the PAYROLL START, STOP, and CANCEL steps.
- ___ 17. When you complete the lab, turn off message revision by issuing the REVISE MSG,OFF command. A message similar to the following one is displayed:
- ```
DSI231I NO 'Message Revision Table' IS ACTIVE
```

## **Answers to Lab 2 Exercise 1 questions**

### **Step 7**

PAYROLL START should generate IEF695I, \$HASP373, IEF403I, VAPL29000A, IEE600I, VAPL29010I, and VAPL29999A messages.

You might also see IEA630I message for the first time that you start the payroll application.

### **Step 8**

PAYROLL STOP should generate IEE600I, VAPL29020I, IEF404I, \$HASP395, and IEA989I messages.

### **Step 9**

PAYROLL CANCEL should generate IEA989I, IEE400I, IEF450I, IEE301I, IEA989I, and \$HASP395 messages.

## Example MRT (VAPMRT1)

The following MRT is an example of VAPMRT1 that is coded for these lab exercises. Your MRT will most likely look different:

```

* - - - - - *
* Sample MRT for NetView for z/OS labs. *
* - - - - - *

UPON (MSGID='$HASP100' | MSGID='$HASP373' | MSGID='$HASP395'
 | MSGID='IEE400I')
 REVISE('N' DISPLAY) ! suppress the message

UPON (MSGID='IEF403I' | MSGID='IEF404I')
 NETVONLY ! send to Netview only

UPON (MSGID='IEA989I' | MSGID='IEE301I' | MSGID='IEE600I')
 REVISE('Y' DELETE) ! suppress, no log, no AUTO

UPON (MSGID='IEF450I')
 Select
 When(jobname='PAYROLL ') ! IEF450I from PAYROLL appl:
 REVISE('CY' COLOR) ! change to yellow
 REVISE(1.* 1 JOBNAME) ! Append jobname to end of text
 Otherwise ! for all other appls:
 REVISE('CW' COLOR) ! change to white
 REVISE(1.* 1 JOBNAME) ! Append jobname to end of text
 End

UPON (MSGID='$HASP250')
 REVISE('Y' DISPLAY) ! unsuppress the message
 REVISE('CR HB' COLOR) ! make msgs red/blinking

UPON (ALWAYS)
 Select
 When(MSGID LEFT 6 = 'VAPL29') ! Only VAPL29* messages
 REVISE('CB HR' COLOR) ! make msgs blue/reverse
 Otherwise ! All other messages
 EXIT ! no change - exit
 End
```



## Exercise 2-2: ASSIGN Command

### Lab exercise objectives

This lab exercise uses the NetView **ASSIGN** command to route messages to NetView tasks, such as operators or automation tasks. At the end of the lab, you should be able to use the **ASSIGN** command to create operator groups and also define authorized receivers of messages.

### Lab exercise instructions

Perform the following steps:

- \_\_\_ 1. Access your system and log on to NetView, following the instructions that your instructor provides.
- \_\_\_ 2. By default, just a few messages are assigned. Issue a **LIST MSG=AUTH** command to verify them. The following messages should be displayed:

```
LIST MSG=AUTH
DSI636I AUTH MESSAGE STRING: 'EZL*'
BNH647I PRIORITY LEVEL: 3
DSI638I PRI (1ST): AUTOAON
DSI642I END OF ASSIGN DISPLAY
```

The message assignment for EZL\* exists if AON is or has been enabled. The assignment does not affect your lab exercise.

Unsolicited messages are processed by the default tasks within NetView: SSIR task for z/OS messages and PPT task for VTAM messages.

- \_\_\_ 3. Issue an **AUTOTBL STATUS** command. Verify that the LAB2TBL automation table is active. If not, activate it with the following command:

```
AUTOTBL MEMBER=LAB2TBL, INSERT LAST
```

- \_\_\_ 4. From TSO, submit the BR14 job from NV390.WORKSHOP.DSIPARM. This action creates unsolicited IEF403I and IEF404I messages.

When the job runs, browse the NetView log and shift to the left by pressing PF10. CNMCSSIR is the task that processes the messages.

- \_\_\_ 5. Assign IEF messages to your NetView operator ID by issuing the following command:

```
ASSIGN MSG=IEF*, PRI=TSCCWxx (where xx is your team number)
```

- \_\_\_ 6. Issue the **LIST MSG=AUTH** command to verify the assignment.

- \_\_\_ 7. Submit the BR14 job again from TSO. The IEF messages are displayed on your NCCF screen. The % next to the message ID indicates that the message was assigned to your operator ID. It also is displayed in the NetView log.

- \_\_\_ 8. Disassociate the message assignment by using the following command:

```
ASSIGN MSG=IEF*,DROP
```

- \_\_\_ 9. Remove the LAB2TBL automation table by using the following command:

```
AUTOTBL REMOVE,NAME=LAB2TBL
```

- \_\_\_ 10. Create an operator group called +TCPOPS and assign two automation operators to by using the ASSIGN command as follows:

```
ASSIGN GROUP=+OMEG OP=(OPER1,OPER2)
```

The following message should be displayed:

```
DSI633I ASSIGN COMMAND SUCCESSFULLY COMPLETED
```

- \_\_\_ 11. Use the LIST command as follows to display the operators in the +OMEG group:

```
LIST ASSIGN=GROUP,GROUP=+OMEG
```

The response should display OPER2 and OPER3 as members of the +OMEG group:

```
DSI180I GROUP ID: +OMEG
BNH647I PRIORITY LEVEL: 3
DSI640I OP(ALL): OPER2 OPER3
DSI642I END OF ASSIGN DISPLAY
```

- \_\_\_ 12. Add the AUTO1 task to the +OMEG group as last and OPER1 to the +OMEG group as first. Issue the following commands:

```
ASSIGN GROUP=+OMEG OP=(AUTO1) ADDLAST
ASSIGN GROUP=+OMEG OP=(OPER1) ADDFIRST
```

Display the operators in the +OMEG group again by issuing the LIST command as follows:

```
LIST ASSIGN=GROUP,GROUP=+OMEG
```

The response should be displayed as follows:

```
DSI180I GROUP ID: +OMEG
BNH647I PRIORITY LEVEL: 3
DSI640I OP(ALL): OPER1 OPER2 OPER3 AUTO1
DSI642I END OF ASSIGN DISPLAY
```

At this point, the +OMEG group contains OPER1 as first in the task list and AUTO1 as last.



**Tip:** If no active operator is in the primary group, the secondary group or operator does not receive any messages.

- \_\_\_ 13. Assign the +OMEG group as the authorized receiver for all CNDL\* (OMEGAMON Subsystem messages) and your NetView operator ID (TSCCWxx) as the secondary receiver. Issue the following command:

```
ASSIGN MSG=CNDL*, PRI=+OMEG, SEC=TSCCWxx
```

Verify the message assignment by issuing the LIST command as follows:

```
LIST ASSIGN=AUTH, MSGID=CNDL*
```

The response should be displayed as follows:

```
DSI636I AUTH MESSAGE STRING: 'CNDL*'
BNH647I PRIORITY LEVEL: 3
DSI638I PRI (1ST): +OMEG
DSI639I SEC (ALL): TSCCW01
DSI642I END OF ASSIGN DISPLAY
```

- \_\_\_ 14. Assign your NetView operator ID (TSCCWxx) as the authorized receiver for the following messages:

Message CNDL001I OMEGAMON SUBSYSTEM V620 INITIALIZED - SSID = CNDL

Issue the following command:

```
ASSIGN MSG=CNDL001I, PRI=TSCCWxx
```

Message CNDL002I OMEGAMON SUBSYSTEM V620 TERMINATED - SSID = CNDL

Issue the following command:

```
ASSIGN MSG=CNDL002I, PRI=TSCCWxx
```

At this point, TSCCWxx is the authorized receiver for CNDL001I and CNDL002I and is also the secondary receiver for all other CNDL\* messages. The +OMEG group is the authorized receiver for all CNDL\* messages.



**Tip:** As discussed in the lecture, more specific assignments, such as CNDL001I, take precedence over less specific assignments, such as CNDL\*.

- \_\_\_ 15. Stop the OMEGAMON subsystem by issuing a P CANACN command from either NetView or the system console.

What did you see on your NetView operator screen? \_\_\_\_\_

\_\_\_\_\_  
\_\_\_\_\_

- \_\_\_ 16. Restart the OMEGAMON subsystem by issuing an S CANACN command. What did you see on your NetView screen?

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

- \_\_\_ 17. Remove all group and message assignments that you created during the lab exercise.

Delete the +OMEG operator group by issuing the following command:

```
ASSIGN GROUP=+OMEG,DROP
```

Delete the assignment for all CNDL\* messages by issuing the following command:

```
ASSIGN MSG=CNDL*,DROP
```

## Answers to Lab 2 Exercise 2 questions

### Step 15

You should see one CNDL message on TSCCWxx with TSCCWxx as the authorized receiver for the CNDL002I message:

```
% CNDL002I OMEGAMON SUBSYSTEM V620 TERMINATED - SSID = CNDL
```

The % indicates that this message is sent to the authorized receiver. Browse DSILOG to see that automation table was driven for the CNDL002I message on TSCCWxx. Look for these messages in DSILOG, and use PF10 to scroll left. TSCCWxx is the task.

```
* 12:39:00 E CNDL030I FUNCTION KCNDLCF STOPPED
* 12:39:00 E CNDL030I FUNCTION DIOPINIT STOPPED
% 12:39:00 E CNDL002I OMEGAMON SUBSYSTEM V620 TERMINATED - SSID =
CNDL
12:45:45 * BR NETLOGA
```

### Step 16

You should also here see several CNDL messages on TSCCWxx:

```
% CNDL001I OMEGAMON SUBSYSTEM V620 INITIALIZED - SSID = CNDL
* CNDL190I SUBSYSTEM ADDRESS SPACE INITIALIZATION ROUTINE
COMPLETED
 SUCCESSFULLY
* CNDL034I SUBSYSTEM START MEMBER KCNSTR00
* CNDL027I FUNCTION KCNDLCF STARTED
* CNDL027I FUNCTION DIOPINIT STARTED
```

The % indicates TSCCWxx is the authorized receiver for message CNDL001I. The \* indicates TSCCWxx is a secondary receiver of the message. Each of the messages flagged as secondary copies also routed to an authorized receiver. For example, if you browse DSILOG, you see that CNDL001I appears twice. CNDL001I appears once for the secondary receiver and once for the authorized receiver.

## Exercise 2-3: NetView and z/OS consoles

### Lab exercise objectives

This lab exercise uses the NetView console management commands to manage EMCS consoles for NetView tasks. At the end of the lab, you should be able to perform the following tasks:

- Use the NetView **DISCONID**, **GETCONID**, and **RELCONID** commands to manage z/OS consoles assigned to NetView tasks.
- Explain how z/OS consoles get assigned to NetView tasks.

### Lab exercise instructions

Perform the following tasks:

1. From NetView AOFDA domain, issue a HELP GETCONID command and read how a console ID associates with a NetView task. What is the default console name if none is specified (GETCONID, CNMSTYLE definitions, and so on)?  

---
2. Issue a DISCONID command to display all consoles for all NetView tasks. You should see messages similar to the following text:

```
* AOFDA DISCONID
' AOFDA
CNM492I OPERATOR ID CONSOLE ID CONSOLE NAME
CNM492I -----
CNM492I TSCCW01 EXTENDED TSC10WAD
CNM492I AUTOAON EXTENDED AUTNOAAD
CNM492I AUTO1 0 * *MASTER*
CNM492I END DISPLAY
```

All tasks use the default console name assignment of the task name.

3. If your operator ID, TSCCWxx, already has a console, issue the RELCONID command to remove any previously obtained consoles. You should see the following response:

```
CNM569I MVS CONSOLE ID RELEASED
```

4. Issue a LIST \* command for your NetView operator. Within the response, you should see a message similar to the following text:

```
DEFAULT MVS CONSOLE NAME: TSC10WAD
```

- \_\_\_ 5. Issue an MVS D A,L command. When that command completes, issue another DISCONID command. You should see a response similar to the following text:

```
* AOFDA DISCONID
' AOFDA
CNM492I OPERATOR ID CONSOLE ID CONSOLE NAME
CNM492I ----- -
CNM492I TSCCW01 EXTENDED TSC10WAD
CNM492I AUTOAON EXTENDED AUTNOAAD
CNM492I AUTO1 0 * *MASTER*
CNM492I END DISPLAY
```

TSCCWxx now has a console associated with it. This was assigned automatically when you issued the MVS D A,L command.

- \_\_\_ 6. Route a MVS D A,L command to the AOFDB domain.



**Tip:** Issue the AOFDB: MVS D A,L command.

You see the result from the display. Issue the DISCONID command first on the AOFDA domain, and nothing new is displayed. However, if you issue the AOFDB: DISCONID command, a new console is allocated on the AOFDB domain for your TSCCWxx user as shown on the following text:

```
* AOFDA AOFDB: DISCONID
* AOFDB DISCONID
' AOFDB
CNM492I OPERATOR ID CONSOLE ID CONSOLE NAME
CNM492I ----- -
CNM492I AUTO1 0 * *MASTER*
CNM492I TSCCW01 EXTENDED TSC10WBD
CNM492I END DISPLAY
```

- \_\_\_ 7. Switch to your z/OS system console. Issue a NetView command from the system console, using the default command designator of %. Issue %LIST \* command at the system console. You should see the following response:

```
STATION: AUTO1 TERM: AUTO1
HCOPY: NOT ACTIVE PROFILE: DSIPROFC
STATUS: ACTIVE IDLE MINUTES: 0
ATTENDED: YES CONS CURRENT COMMAND: LIST
AUTHRCVR: NO CONTROL: GLOBAL
NGMFADMN: NO DEFAULT MVS CONSOLE NAME: AUT1OTAD
NGMFVSPN: NNNN (NO SPAN CHECKING ON NMC VIEWS)
NGMFCMDS: YES AUTOTASK: YES
IP ADDRESS: N/A
OP CLASS LIST: NONE
DOMAIN LIST: AOFDA (I) AOFDB (I)
ACTIVE SPAN LIST: NONE
Task Serial: 604 REXX Environments: 2 (1%)
Messages Pending: 0 Held: 0
WLM Service Class: Not Available
END OF STATUS DISPLAY
```

Why did the LIST command run under AUTO1? \_\_\_\_\_

\_\_\_\_\_



## Answers to Lab 2 Exercise 3 questions

### Step 1

The default console name is the name of the task that issues the MVS command, for example, TSCCWxx.

### Step 6

To control the creation of unique console names, you can perform one of the following tasks:

- Issue a RELCONID command before each MVS command to free up the console name.
- Issue a GETCONID command to specify a unique console name for each task.
- Edit CNMSTYLE and define a console mask with the ConsMask statement.

### Step 7

Commands from the system console run under the AUTO1 task because it is associated with the *master* console as displayed by the following text:

```
AUTOTASK.?Master.Console = *ANY*
function.autotask.Master = AUTO1 // autotask with master console
```



# Student exercises for Unit 3

.....

## Lab exercise overview

In this exercise, you learn to route commands to other NetView tasks within the same domain or in other NetView domains. You also learn to create and manage NetView timers.

## Exercise 3-2: Routing commands to other tasks and domains

### Lab exercise objectives

This lab introduces you to the NetView facilities to route commands to tasks within the same NetView domain or other NetView domains. At the end of the lab, you should be able to perform the following tasks:

- Use EXCMD to route commands to another task in the same NetView domain.
- Use RMTCMD to route commands to tasks in other NetView domains.
- Use labelled commands for both EXCMD and RMTCMD.

### Lab exercise instructions

Perform the following steps:

- \_\_\_ 1. Access your system and log on to the NetView AOFDA domain, following the instructions that the instructor provides.
- \_\_\_ 2. Issue the LIST \* command on task AUTO1 by entering the following command:  

```
EXCMD AUTO1,LIST *
```

The output of the LIST command is not displayed, just a message that the EXCMD command ran. To see the actual output of the LIST command, browse the NetView log.
- \_\_\_ 3. You can send a command to another task and see the output of the command by using a labelled command as follows:  

```
/AUTO1: LIST *
```

- \_\_\_ 4. The RMTCMD command routes commands to another NetView domain, creating a distributed autotask on the remote NetView that uses your operator ID (by default). Because you are starting an autotask, no password is required.

Use RMTCMD to establish a connection from AOFDA to AOFDB domains by issuing the following command:

```
RMTCMD LU=AOFDB,LIST *
```

The output of the LIST \* command shows that you are a distributed autotask. How can you determine that? \_\_\_\_\_

\_\_\_\_\_

- \_\_\_ 5. The RMTCMD command supports communication over SNA and TCP/IP. Using the online help, can you determine whether your RMTCMD was using SNA or TCP/IP?
- \_\_\_\_\_

- \_\_\_ 6. You can use a labelled command as follows to shorten the syntax of RMTCMD:

```
AOFDB: LIST *
```

- \_\_\_ 7. End your RMTCMD session by issuing the following command:

```
AOFDB: LOGOFF
```

## Answers to Lab 3 Exercise 2 questions

### Step 4

In the response to the LIST command, the STATION name and TERM name are the same:

```
STATION: TSCCWxx TERM: TSCCWxx
```

### Step 5

When you specify the RMTCMD command, you must choose between using the LU or the DOMAIN parameters. If you use the LU parameter, you request an SNA session. If you use the DOMAIN parameter, you request a TCP/IP session. If your NetView domain supports both SNA and TCP/IP for RMTCMD, NetView attempts to start the TCP/IP session first if you specify a labelled RMTCMD.

Each NetView contains definitions in CNMSTYLE for controlling SNA and IP access. Browse CNMSTYLE and find the RMTINIT statement. Read the comments for further information.

You can also issue a RMTCMD QUERY RMTDOMS command to determine the active RMTCMD sessions and the communication method. An example follows:

```
* AOFDA RMTCMD QUERY RMTDOMS
' AOFDA
BNH060I RMTCMD QUERY INFORMATION
BNH061I -----
BNH068I REMOTE NETVIEW VERSION TRANSPORT
BNH061I -----
BNH069I USIBMES.AOFDB V6R1 TCP/IP
BNH690I IP ADDRESS: 10.31.187.195 PORT: 4022
```

## Exercise 3-3: Timer commands

### Lab exercise objectives

This lab introduces you to the NetView facilities for scheduling and managing timers in a single NetView domain or multiple NetView domains. At the end of the lab, you should be able to perform the following tasks:

- Schedule AT, EVERY, and AFTER timers in NetView.
- Use the LIST TIMER command to display NetView timers.
- Use the TIMER panel interface to manage NetView timers, including timers in another NetView domain.

### Lab exercise instructions

Perform the following steps:

- \_\_\_ 1. Issue the following AFTER command to schedule a NetView timer to run the MYCLIST command after one minute has passed:

```
AFTER 00:01:00,MYCLIST
```

- \_\_\_ 2. Issue the following LIST TIMER command to verify that the timer has been set:

```
LIST TIMER=ALL
```

The response should resemble the following text:

```
* AOFDA LIST TIMER=ALL
' AOFDA
DISPLAY OF OUTSTANDING TIMER REQUESTS
TYPE: AFTER TIME: 07/13/11 13:50:59
 COMMAND: MYCLIST
 OP: TSCCW01 (TSCCW01) ID: SYS00013 TIMEFMSG: NO
GMT
1 TIMER ELEMENT(S) FOUND FOR TSCCW01
END OF DISPLAY
```

Wait one minute. When the AFTER timer opens, you should see messages similar to the following text:

```
- AOFDA DSI208I TIME EXPIRATION - ID= 'SYS00013' - CMD=
'MYCLIST'
C AOFDA Hello from MYCLIST running under task TSCCW01
```

The DSI208I message is displayed because of the AFTER timer. The second message is displayed because of the MYCLIST command.

- \_\_\_ 3. Schedule an AFTER timer as follows to issue MYCLIST under AUTO1 after one minute has passed:

```
AFTER 00:01:00,ROUTE=AUTO1,MYCLIST
```

- \_\_\_ 4. Use the LIST TIMER command as follows to verify that the timer has been set:

```
LIST TIMER=ALL,OP=AUTO1
```

You need to specify the OP parameter to display the timer for AUTO1.

- \_\_\_ 5. When one minute has passed, browse the NetView log to verify that AUTO1 issued the MYCLIST command.

- \_\_\_ 6. Schedule an EVERY timer command as follows to issue MYCLIST under your operator ID every thirty seconds, giving the timer an ID of MYTIMER:

```
EVERY 00:00:30,ID=MYTIMER,MYCLIST
```

- \_\_\_ 7. When the timer has run twice, issue the PURGE TIMER command as follows to remove the timer:

```
PURGE TIMER=MYTIMER
```

- \_\_\_ 8. Schedule an AT timer command for MYCLIST to run under your operator ID at a time that is *30 minutes from now*. Be sure to save the timer to the Save/Restore database. An example follows:

```
AT 10:30:00,SAVE,MYCLIST
```

- \_\_\_ 9. Issue the LIST TIMER command as follows to verify that the timer has been set and saved:

```
LIST TIMER=ALL
```

- \_\_\_ 10. Issue the TIMER command to open a full-screen panel interface to the NetView timer commands. You should see the *Timer Management* panel (EZLK6000) with several timers scheduled by NetView. *Do not modify or delete any of the NetView timers.*

- \_\_\_ 11. The cursor should be positioned next to the first timer in the list. Select option one (add) to create a new timer.

- \_\_\_ 12. Change the Timer Type to be an AFTER timer and press Enter. You should see the Set AFTER Timer panel (EZLK6130).

- \_\_\_ 13. Fill in the appropriate fields to schedule an AFTER timer to run the MYCLIST command under the AUTO1 task in 30 seconds. Press Enter to create the timer.

You should see a message similar to the following text:

```
EZL973I REQUESTED TIMER SYS00098 ADDED ON AOFDA
```

Press PF3 to return to the Timer Management panel.

- \_\_\_ 14. After 30 seconds have elapsed, press PF5 to refresh the Timer Management panel. The AFTER timer should no longer be displayed.

Why were you not interrupted when the AFTER timer ran? \_\_\_\_\_

- \_\_\_ 15. Select option one (add) to create another timer. Define an EVERY timer to run the MYCLIST command every 30 seconds on your operator ID. Specify a timer ID of MYTIMER.



**Tip:** If you are not on the Set EVERY timer panel (EZLK6110), change the Timer Type field for an EVERY timer and press Enter. Also, you must specify a day (1 through 9). Select option nine and keep the default of 000 days to schedule the timer today.

Press PF3 when you have defined the EVERY timer.

- \_\_\_ 16. Press PF6 to ROLL to the NCCF screen. Each time MYTIMER opens, a message similar to the following text is displayed:

```
Hello from MYCLIST running under task TSCCWxx
```

- \_\_\_ 17. Press PF6 to ROLL back to the Timer Management panel. Tab over to MYTIMER and select option two (display/change) to edit the timer. You should see the Set EVERY Timer panel (EZLK6110).

- \_\_\_ 18. Change the timer type from EVERY to AT, and specify a time one minute from now. Press Enter to save the timer.

You should see message EZL974I REQUESTED TIMER MYTIMER CHANGED ON AOFDA.

Press PF3 to return to the Timer Management panel.

- \_\_\_ 19. Press PF6 to ROLL to the NCCF screen and wait for MYTIMER to open. A message similar to the following text is displayed:

```
Hello from MYCLIST running under task TSCCWxx
```

- \_\_\_ 20. After MYTIMER runs, press PF6 to roll back to the Timer Management panel. Press PF5 to refresh the timers. MYTIMER should no longer be displayed.

- \_\_\_ 21. Type a question mark (?) in the Target field and press Enter. The Remote Target Selection panel (EZLK5500) should display the AOFDA and AOFDB domains, along with session information for each.

- \_\_\_ 22. Select the AOFDB domain and press Enter. You should now see timers that are scheduled to run in the NetView AOFDB domain.



Several fields are now filled in as follows:

- The IP Addr and Port fields show that the TIMER function is using RMTCMD over TCP/IP to communicate with the AOFDB domain.
- The Remote Target Date and Time field is now filled in. In some cases, the target domain might not be in the same time zone.

\_\_\_ 23. Press PF3 to close the Timer Management panel.

\_\_\_ 24. Issue the TIMER command again. You should see the Timer Management panel with timers for your local NetView domain. By default, the TIMER command always displays timers in your local NetView domain. You can view timers in remote NetView domains if you specify a TARGET parameter on the TIMER command. For more information, see the help for the TIMER command.

## Answers to Lab 3 Exercise 3 questions

Step 14

Because the AFTER timer was scheduled under AUTO1, you do not see any messages on your operator ID.

## Exercise 3-4: Command Revision Table (CRT)

### Lab exercise objectives

This lab introduces you to NetView Command Revision Table (CRT). At the end of the lab, you should be able to perform the following tasks:

- Install the necessary code for activating the CRT.
- Explain how to use the CRT to revise commands.
- Modify the CRT to achieve the required action for a specific command.

### Lab exercise instructions

Perform the following steps:

1. To enable the Command Revision Table, some items must be in place. Issue MVS D PROG,LNKLST to verify that NETV610.SCNMLNKN is in your Linklist. This library contains the DSIRCVEX module, which must be in the Linklist. This step has been done for you to prepare the system.
2. Modify the MPFLST to include the user exit that handles the command revision. This step has been prepared for you also. Issue MVS D MPF and verify user exit DSIRVCEX is installed.

With these functions in place, you are now ready to start working with the CRT.

3. Check that no CRT is active by issuing REVISE STATUS. Activate the default CRT CNMSCRT1 by issuing REVISE CMD, MEMBER=CNMSCRT1. You should receive the following response:

```
BNH608I 'LOAD CMD CNMSCRT1' request completed successfull
```

4. Test your loaded CRT by issuing MVS SEND 'THIS IS INTENDED FOR ALL' to receive the following response:

```
* AOFDA MVS SEND 'THIS IS INTENDED FOR ALL'
E AOFDA TLH447E Please do not broadcast to all.
```

This is the CRT taking control, and the command is stopped. If you issue MVS SEND 'ONLY FOR TSCCWxx' USER=TSCCWxx instead, the command is passed for execution.

5. Browse CNMSCRT1 and look at the sample logic.

- \_\_\_ 6. Look at some more logic with the CRT. The system comes with both a sample CRT CNMSCRT1, which we looked at previously. The same sample includes a REXX procedure as follows:

```
UPON (CMDVERB = 'V' ! CMDVERB = 'VARY')
 SELECT
 WHEN (WORD 2 = 'NETVIEW') ! V
 NetView, (anytext)
 NETVONLY=CNMSRVMC ! handle in NetView
 OTHERWISE ! all other VARY cmds untouched
 END
```

The NETVONLY=CNMSRVMC calls the REXX procedure CNMSRVMC, which is a provided sample. In such a REXX procedure, you can create your own logic to decide whether or not to issue the command.

- \_\_\_ 7. Copy the CNMSCRT1 to your NV390.WORKSHOP.DSIPARM and modify it so that a STOP or P command on procedure CANACN calls REXX CRT1. Browse CRT1 to see what occurs if you try to do STOP CANACN. CRT1 is modified from CNMSCRT1. Several REXX procedures can be called by different NETVONLY= statements.
- \_\_\_ 8. When you have changed your CNMSCRT1, activate it. Stop the active copy (REVISE CMD,OFF) then activate it with REVISE CMD, MEMBER=CNMSCRT1.
- \_\_\_ 9. Test your new CRT by issuing MVS P CANACN. The response should be as follows:

```
* AOFDA MVS P CANACN
> AOFDA 12 TLH916W OMEGAMON Subsystem should not be stopped
 before the TEMS. Answer "Y" to continue
```

```
Reply to the WTOR with N
E AOFDA IEE600I REPLY TO 12 IS;N
E AOFDA TLH917I STOP CANACN CANCELED.
```

- \_\_\_ 10. Stop the active CRT by issuing REVISE CMD,OFF.

## Answers to Lab 3 Exercise 4 questions

### Step 7

The following entries should be incorporated into the CNMSCRT1 member:

```
* UPON statements must follow control statements
UPON (CMDVERB = 'P' | CMDVERB = 'STOP')
 SELECT
 WHEN (WORD 2 = 'CANACN') ! P CANACN
 NETVONLY=CRT1 ! handle in NetView
 OTHERWISE ! all other VARY cmds untouched
 END
```



# Student exercises for Unit 4

.....

## Lab exercise overview

In this exercise, you log on to NetView to manage the automation tables with the AUTOTBL and AUTOMAN commands.

## Exercise 4-1: Managing the NetView automation tables

### Lab exercise objectives

This lab introduces you to managing the NetView automation table. At the end of the lab, you should be able to perform the following tasks:

- Use the AUTOTBL command to display the active automation table and load additional automation tables.
- Use the AUTOMAN panel interface to manage the automation tables.

### Lab exercise instructions

Perform the following steps:

1. Display the currently active automation table by issuing the AUTOTBL STATUS command. You should see a response similar to the following text:

```
* AOFDA AUTOTBL STATUS
' AOFDA
BNH361I THE AUTOMATION TABLE CONSISTS OF THE FOLLOWING LIST OF
MEMBERS:
AOFDAPPT COMPLETED INSERT FOR TABLE Ä1: DSITBL01 AT 11/10/10
08:44:14 (FIRST)
AOFDAPPT COMPLETED INSERT FOR TABLE Ä2: VAPLAT AT 11/10/10
08:44:14
```

Two automation tables are active: DSITBL01 and VAPLAT. Automation tables can be loaded by using the AUTOTBL command or, as in this case, when NetView initializes. Automation tables that load during initialization are specified in CNMSTYLE with the AUTOCMD.statement.

- \_\_\_ 2. Browse the CNMSTYLE member. On the command line (CMD==>), specify ALL 1.8 AUTOCMD. and press Enter. The trailing period is necessary. Running this command displays all of the AUTOCMD.statements in CNMSTYLE and its included members.

You should see several AUTOCMD. statements similar to the following text:

```
AUTOCMD.DSITBL01.list = ListName
AUTOCMD.DSITBL01.marker = (AON)
AUTOCMD.DSITBL01.order = *FIRST*
AUTOCMD.VAPLAT.MARKER = VPL
AUTOCMD.VAPLAT.ORDER = B
```

Automation tables are loaded based on the *order* statement in CNMSTYLE. DSITBL01 loads first by default (\*FIRST\*). All other automation tables load based on an EBCDIC sort of the *order* statement. VAPLAT is a user-defined automation table that loads second. In this case, there are no other automation tables.

Suppose you were asked to define AUTOCMD.statements for inserting an automation table between DSITBL01 and VAPLAT. What would your AUTOCMD.statements look like?

\_\_\_\_\_

\_\_\_\_\_

- \_\_\_ 3. Issue an AUTOTBL command to insert the LAB4TBL as the first automation table by issuing the following command:

```
AUTOTBL MEMBER=LAB4TBL, INSERT FIRST
```

The AUTOTBL request should fail, and respond with the following message:

```
BNH367E UNABLE TO COMPLETE AUTOTBL INSERT REQUEST. REASON CODE:
104
```

Why does the insert initially fail? \_\_\_\_\_

\_\_\_\_\_

- \_\_\_ 4. Insert LAB4TBL as the second automation table by issuing the following command:

```
AUTOTBL MEMBER=LAB4TBL, INSERT AT=2
```

The following BNH360I message is displayed:

```
BNH360I INSERT REQUEST COMPLETED FOR DSIPARM MEMBER LAB4TBL AT
LOCATION 2 WITHIN THE LIST OF ACTIVE AUTOMATION TABLES
```



- \_\_\_ 5. Issue the AUTOMAN command to display the active automation tables in a panel interface. You should see the Automation Table Management panel (EZLK8500) as follows:

```
EZLK8500 AUTOMATION TABLE MANAGEMENT

AUTOMATION TABLE Enter any character in the selection fields
SEL POS NAME STATUS MARKERS TASK DATE TIME
- - - - - - - -
- 1 DSITBL01 ENABLED (FIRST) (AON) AOFDAPPT 11/10/10 08:44:14
- 2 LAB4TBL ENABLED TSCCW01 11/10/10 08:52:46
- 3 VAPLAT ENABLED AOFDAPPT 11/10/10 08:44:14

Command ==>
F1=Help F2=Main Menu F3=Return F4=Commands F5=Refresh F6=Roll
F7=Backward F8=Forward F9=Responses F10=Global Commands F12=Cancel
```

This panel displays the three automation tables that are active with LAB4TBL inserted as the second table.

- \_\_\_ 6. Type any non-blank character in the SEL column next to LAB4TBL and press PF4 or ENTER. This action displays a list of commands that can be issued against LAB4TBL. A window opens on the panel with a list of nine commands. From here, you can enable or disable, test the table, and so on.
- \_\_\_ 7. Select option nine (Display options) to view the list of display options.
- \_\_\_ 8. Select option one (Browse table with includes). The NetView browse function displays the source of the LAB4TBL automation table. The comments indicate one that LAB4TBL is currently *under construction*. Press PF3 to return to AUTOMAN.
- \_\_\_ 9. If LAB4TBL does not contain any automation logic, delete it from the list of active automation tables. Press PF3 until you return to the command window. Issue Unload LAB4TBL, option seven. The Automation Table Management panel (EZLK8500) displays a message as follows, indicating the unload was successful:
- ```
EZL919I ALL ACTIONS SUCCESSFULLY COMPLETED
```
- ___ 10. Press PF3 to end AUTOMAN. You use AUTOMAN again to load automation tables in the lab exercises for Unit 5.

Answers to Lab 4 Exercise 1 questions

Step 2

Suppose you were asked to insert an automation table named MYTABLE between DSITBL01 and VAPLAT when NetView initializes. The CNMSTYLE statements should look similar to the following text:

```
AUTOCMD.mytable.MARKER = MYT  
AUTOCMD.mytable.ORDER  = A
```

Step 3

The insert fails because DSITBL01 is locked as the first automation table with an AUTOCMD.DSITBL01.*order* statement in CNMSTYLE, as the following text shows:

```
AUTOCMD.DSITBL01.order = *FIRST*
```

The help for message BNH367E return code 104 states as follows:

```
A request to establish a table as the first table  
within the list of automation tables failed because  
another table is already established as the locked  
first table. The table must be removed before similar  
attempts are successful. See the description of the  
AUTOTBL command for more information about the FIRST  
keyword.
```

Student exercises for Unit 5

.....

Lab exercise overview

In this exercise, you code automation table statements to trap and automate messages that indicate the status of resources. You define automation to run commands on automation tasks.

Data sets used in this lab

NV390.WORKSHOP.DSIPARM _____

Exercise 5-1: Coding automation table statements

Lab exercise objectives

This lab provides an in-depth look at coding the NetView automation table statements. At the end of the lab, you should be able to code automation table entries to perform the following tasks:

- Trap messages.
- Take actions when the messages occur.

Lab exercise instructions

This exercise is about defining and starting automation operators. Perform the following steps:

1. Log on to TSO and edit the **LABOPFU** member in NV390.WORKSHOP.DSIPARM. Create NetView operator definitions for several automation tasks: MVSAUTO, TSOAUTO, RMFAUTO, and MISCAUTO

For each operator definition, use definitions similar to *TSCCW01*, replacing it with the automation task name:

```
TSCCW01 OPERATOR      PASSWORD=TSCCW01
          PROFILEN      DSIPROFA
```

- ___ 2. From NetView AOFDA domain, issue the following REFRESH command to dynamically enable the new operator definitions:

```
REFRESH OPERS
```

You should see the following messages:

```
DWO831I OPERATOR MVSAUTO IS ADDED TO THE GROUP OF VALID NETVIEW
OPERATORS
DWO831I OPERATOR TSOAUTO IS ADDED TO THE GROUP OF VALID NETVIEW
OPERATORS
DWO831I OPERATOR RMFAUTO IS ADDED TO THE GROUP OF VALID NETVIEW
OPERATORS
DWO831I OPERATOR MISCAUTO IS ADDED TO THE GROUP OF VALID NETVIEW
OPERATORS
DSI633I REFRESH COMMAND SUCCESSFULLY COMPLETED
```

If you do not see any messages from the REFRESH OPERS command, LABOPFU loaded because of the memStore function. Unload LABOPFU with the MEMSTOUT command and run REFRESH OPERS again as follows:

```
MEMSTOUT UNLOAD DSIPARM.LABOPFU
REFRESH OPERS
```

- ___ 3. Issue an AUTOTASK to log MVSAUTO on as an automation task. An example follows:

```
AUTOTASK OPID=MVSAUTO
```

You should see the following response:

```
CNM570I STARTING AUTOMATION TASK MVSAUTO
DSI530I 'MVSAUTO' : 'OST' IS READY AND WAITING FOR WORK
```

Issue an AUTOTASK for the three remaining automation tasks that you just defined.

Lab exercise instructions

This exercise is about coding automation table statements, Perform the following steps:

- ___ 4. Switch to your TSO session and edit the LAB5TBL member in NV390.WORKSHOP.DSIPARM. You create automation table statements to trap specific messages and run commands to restart applications, such as TSO, RMF, and so on.



Note: Type “CAPS OFF” on the TSO command line before you begin creating your automation table statements. With this setting, you can use mixed-case text in the automation table; improving its readability.

Create automation statements as follows:

- IEF404I automation: (generic message as follows, indicating application has ended)

```
IEF404I jobname - ENDED - TIME=14.17.41
```

- ___ a. Create automation to trap the IEF404I and restart the failed job (second token, *jobname*) automatically. Route the command to MVSAUTO.

- ___ b. Log the message in NetView.

- ___ c. Suppress the message from the system log.

- IEF403I automation: (generic message as follows, indicating application now active)

```
IEF403I jobname - STARTED - TIME=14.27.36
```

- ___ a. Create automation to trap the IEF403I and call the APPLUP EXEC with the *jobname* (second token). Route the command to MVSAUTO.

- ___ b. Log the message in NetView.

- ___ c. Suppress the message from the system log.

- IKT010D automation: (TSO users active message with reply, as follows)

```
nn IKT010D 00001 USER(S) ACTIVE, REPLY 'SIC' OR 'FSTOP'
```

- ___ a. Create automation to trap the IKT010D message and reply automatically with FSTOP. Route the reply to the TSOAUTO task.

- ___ b. Log the message in NetView.

- ___ c. Suppress the message from the system log.

TSO restarts when the IEF404I message is issued.

- CNDL001I automation: (OMEGAMON Subsystem Initialized message as follows)

```
CNDL001I OMEGAMON SUBSYSTEM V620 INITIALIZED - SSID = CNDL
```

- ___ a. Create automation to trap the CNDL001I message and call the APPLUP EXEC with the SSID name from the message.

- ___ b. Log the message in NetView.

- ___ c. Suppress the message from the system log.

- ___ 5. When finished creating the automation table statements in LAB5TBL, press PF3 to save your work.

Lab exercise instructions

This exercise is about testing your automation table statements, Perform the following steps:

- ___ 6. In NetView AOFDA, use the following AUTOTBL command to test the syntax of your LAB5TBL:

```
AUTOTBL MEMBER=LAB5TBL, TEST
```

You should see the CNM501I message. If not, correct the errors and issue the AUTOTBL command to test LAB5TBL again. The response should be similar to the following text:

```
CNM501I TEST OF NETVIEW AUTOMATION FILE "LAB5TBL" WAS SUCCESSFUL
```

- ___ 7. When your automation table test is successful, load it as the second table in the list, using the AUTOMAN panel interface.

- ___ 8. When LAB5TBL loads correctly, use your system console session to stop several of the applications as follows to test your automation:

```
P RMF
P CANACN
P AUTOSI
```

- ___ 9. Browse DSILOG and look for CNM493I messages pertaining to each application that stopped. For example, for the RMF application, you should see messages similar to the following text:

```
CNM493I LAB5TBL : #0000023 : MVS START RMF
IEF404I RMF - ENDED - TIME=15.49.09
```

RMF has ended. Automation for IEF404I has parsed the jobname, RMF, from the failure message and issued the MVS START RMF command as follows.

```
CNM493I LAB5TBL : #0000036 : APPLUP RMF
IEF403I RMF - STARTED - TIME=15.49.09
```

RMF has started. Automation for IEF403I has parsed the jobname, RMF, and passed it as a parameter to the APPLUP EXEC as follows.

```
DSI039I MSG FROM MVSAUTO : JOB NAME RMF IS NOW ACTIVE.
```

The APPLUP EXEC issues the NetView MSG command to inform operators that RMF is now active.

- ___ 10. When all other automation works successfully, issue P TSO to bring TSO down while you are still logged on. Running the command tests the automation table statements that you created for TSO.

You should see messages similar to these in DSILOG. Your automation should parse the reply number (84) and respond with FSTOP.

```
- CNM493I LAB5TBL : #0000049 : MVS REPLY 84,FSTOP
> 84 IKT010D 00001 USER(S) ACTIVE, REPLY 'SIC' OR 'FSTOP'
E IEE600I REPLY TO 84 IS;FSTOP
```

What does the #00000049 in the example CNM493I mean? _____

You can use the CNM493I message as an audit trail of the automated actions taken when an event is received. This can be very helpful when you initially create your automation table statements.



Tip: After your automation is in place, you can turn off the logging of CNM493I by issuing the NetView OVERRIDE or DEFAULTS commands. Or you can modify CNMSTYLE with a DEFAULTS.CNM493I={YES | NO} statement.

11. Issue the AUTOMAN command again. Select your automation table, LAB5TBL, and press PF4 (or ENTER) to display the command window.

Select option nine (DISPLAY options), which opens another window with a list of display selections. Select option seven (display AUTOCNT statistics).

You should see a panel similar to the following graphic, which is an example of the AUTOCNT STATISTICS display for LAB5TBL:

CNMKWIND OUTPUT FROM AUTOCNT STATISTICS									
Top of Data									
DW0800I AUTOMATION TABLE MSG DETAIL REPORT BY TSCCW01									
----- (LAB5TBL /AUTOM728 MESSAGE DETAILS 11/11/10 03:30:15) -----									
TSCCW01 COMPLETED INSERT FOR TABLE Ä2: LAB5TBL AT 11/11/10 03:20:58									
!-- PERCENTAGES --!									
STMT	L	SEQUENCE	NUMBER/	MEMBER	COMPARE	MATCH	E C A D	MATCH/	COMP/
NUMB	I	LABEL	NAME	NAME	COUNT	COUNT	C I I I	COMP	TOTAL

MEMB				LAB5TBL					
0003	S	Ä0000012		LAB5TBL	147	4	0	2.7	100.0
0004	S	Ä0000014		LAB5TBL	4	0	0	0.0	2.7
0005	S	Ä0000015		LAB5TBL	4	0	0	0.0	2.7
0006	S	Ä0000016		LAB5TBL	4	0	0	0.0	2.7
0007	S	Ä0000017		LAB5TBL	4	0	0	0.0	2.7
0008	S	Ä0000018		LAB5TBL	4	0	0	0.0	2.7
0009	S	Ä0000019		LAB5TBL	4	0	0	0.0	2.7
0010	S	Ä0000020		LAB5TBL	4	0	0	0.0	2.7
0011	S	Ä0000021		LAB5TBL	4	0	0	0.0	2.7
0012	S	Ä0000022		LAB5TBL	4	1	0	25.0	2.7
0013	S	Ä0000023		LAB5TBL	3	2	2	66.7	2.0
0014	S	Ä0000029		LAB5TBL	1	1	1	100.0	0.7
0016	S	Ä0000036		LAB5TBL	143	4	1	2.8	97.3
0017	S	Ä0000042		LAB5TBL	139	0	1	0.0	94.6
0018	S	Ä0000049		LAB5TBL	139	1	1	0.7	94.6
0019	S	Ä0000057		LAB5TBL	138	0	1	0.0	93.9

DW0800I AUTOMATION TABLE MSU DETAIL REPORT BY TSCCW01									
----- (LAB5TBL /AUTOM728 MSU DETAILS 11/11/10 03:30:15) -----									
NO MSU STATEMENTS IN CURRENT AUTOMATION TABLE									
TO SEE YOUR KEY SETTINGS, ENTER 'DISPFK'									
CMD==>									

For example, line number twelve was compared 147 times and resulted in 4 matches. Line number twelve is the following text:

```
IF MSGID = 'IEF404I' then
```

Four IEF404I messages matched in LAB5TBL.

Answers to Lab 5 Exercise 1 Questions

Step 10

Within the example CNM493I message, the #00000049 indicates the statement that matches in the automation table. The statement can be a line number or a sequence number. In this case, it is a relative line number (because of the # as the first character) within the automation table member, LAB5TBL. The online help for CNM493I describes this field as follows:

seqnum: The 8-character sequence number field of the first record of the automation statement that generated this command.

Note: If this message is displayed for a statement that was added without sequence numbers, then seqnum will be the line number where the statement was found in the automation table member. The number will be in the format #nnnnnnn.



IBM Tivoli NetView for z/OS 6.1: REXX Programming

Student Exercises

Course: TZ223 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

.....

Student exercises for Unit 1

Lab exercise overview	1-1
Data sets used in this lab	1-1
Exercise 1-1: Simple REXX EXECs	1-1
Lab exercise objectives	1-1
Lab exercise instructions	1-1
Answers to lab exercise 1 questions	1-5
Exercise 1-2: Global variables	1-8
Lab exercise objectives	1-8
Lab exercise instructions	1-8
Answers to lab exercise 2 questions	1-13
Exercise 1-3: Global variables tracing	1-14
Lab exercise objectives	1-14
Lab exercise instructions	1-14
Answers to lab exercise 3 questions	1-15
Exercise 1-4: TRAP and WAIT	1-16
Lab exercise objectives	1-16
Lab exercise instructions	1-16
Answers to lab exercise 4 questions	1-19
Exercise 1-5: EXECIO	1-20
Lab exercise objectives	1-20
Lab exercise instructions	1-20

Student exercises for Unit 1

.....

Lab exercise overview

In these exercises, you create several REXX EXECs in TSO and test them in NetView®. You should have two PCOMM sessions for this lab exercise. One session is for NetView domain AOFDA and the other is for TSO.

Data sets used in this lab

NV390.WORKSHOP.CNMCLST_____

Exercise 1-1: Simple REXX EXECs

Lab exercise objectives

This lab provides an introduction to programming REXX EXECs for NetView. At the end of this lab, you should be able to perform the following tasks:

- Parse EXEC input.
- Utilize several NetView built-in functions.
- Display messages to operators.

Lab exercise instructions

- ____ 1. Access your system and log on to TSO. Follow the instructions that the instructor provides. Log on to NetView AOFDA domain, also following the instructions.

Optionally, issue HILITE REXX on the TSO edit command line. This command modifies the display to use different colors as you create the EXEC. If you do not like the colors, issue HILITE OFF.

- ___ 2. Create a new REXX EXEC, called MYVARS, to perform the following steps:
 - ___ a. Parse the two following inputs to the EXEC:
 - Use the REXX PARSE statement to parse four parameters with the fourth being the rest of the input string.
 - Use the NetView MSGVAR(*n*) EXEC to parse four parameters.
 - ___ b. Use several SAY instructions to display the input parameters for each method of parsing.
 - ___ c. When you finish, save your work.

- ___ 3. Issue the MYVARS EXEC from AOFDA, using more than 4 parameters, as follows:

```
MYVARS parm1 parm2 parm3 parm4 parm5
```

Use lower-case text for the parameters. Your output should look similar to the following text:

```
MYVARS: Input parsed with PARSE ARG
        Parameter 1 is  PARM1
        Parameter 2 is  PARM2
        Parameter 3 is  PARM3
        Parameter 4 is  PARM4 PARM5
MYVARS: Input parsed with NetView MSGVAR(n)
        Parameter 1 is  PARM1
        Parameter 2 is  PARM2
        Parameter 3 is  PARM3
        Parameter 4 is  PARM4
```

Why does the PARSE ARG display two parameters as the last parameter?

Why does the MSGVAR(*n*) ignore the fifth parameter? _____

Why does the EXEC display the parameters in upper case? _____

- ___ 4. To keep the parameters in mixed-case text, prefix the EXEC with NETVASIS. For example, enter text as follows:

```
NETVASIS MYVARS parm1 parm2 parm3 parm4 parm5
```

The input parameters should appear exactly as you typed them.

- ___ 5. Issue the MYVARS EXEC with commas (,) as delimiters between each parameter as follows:

```
NETVASIS MYVARS parm1,parm2,parm3,parm4,parm5
```

What changes in the output? Why? _____

- ___ 6. Update the MYVARS EXEC to query several NetView built-in functions and write their values to the operator:
 - ___ a. Query the following functions:
 - CurSys()
 - Sysplex()
 - MVSlevel()
 - NetView('T')
 - Tower('*')
 - Netid()
 - Domain()
 - Opid()
 - Task()
 - Curconid()
 - Lu()
 - ___ b. Add the use of several REXX functions as follows:
 - Date(): Displays the date in the European format.
 - Date('W'): Displays the current day.
 - Time(): Displays the date in the 24-hour clock format.
 - Words(): Displays the **number** of towers that are enabled.
 - ___ c. Modify the input parameter parsing:
 - ___ d. Test for the existence of input parameters.
 - ___ e. If there are no input parameters, do not bother to use the PARSE ARG and MSGVAR statements.



Tip: Use an IF-THEN construct with a DO-END grouping.

- ___ 7. Save your work when you finish. Issue the MYVARS EXEC without any input parameters. The response should be similar to the following text:

```
Current sysplex name: PLEX12
Current system name : MVSA
Current z/OS level: SP7.1.2
NetView level: Tivoli NetView for z/OS V6R1
Towers enabled: NPDA NLDM TCPIP COLLECT          TEMA IPMGT
NVS OA DISCOVERY
There are 5 towers enabled.
The third tower is: TCPIP COLLECT
NetView domain ID: AOFDA
NetView operator ID: TSCCW01
NetView task type: OST
Current z/OS console name:
Current VTAM LU name: ESIP1017
Current VTAM Network ID: USIBMES
Current date is: 06/15/11
Current time is: 06:01:06
Today's day is: Wednesday
```

If you receive any error messages, correct your coding in MYVARS and retry.

- ___ 8. Edit MYVARS again to prompt for input (using PARSE PULL) from the operator before continuing. For example, ask the operator if the day name was parsed correctly.
- ___ a. Issue a message to the operator identifying the desired input as follows, for example.
- ```
Is the day correct? Enter "GO Y" or "GO N"
```
- \_\_\_ b. Use PARSE PULL to pause the EXEC until the operator replies.
- \_\_\_ c. After a reply is provided, display the response back to the operator. Optionally, test if the response is a Y. If not, display the message again and wait for operator response again.
- \_\_\_ d. Issue another message indicating that the lab exercise is done.



## Answers to lab exercise 1 questions

### Step 3

PARSE ARG displays the last two input parameters as one because of the way you were asked to code the parse:

```
parse arg p1 p2 p3 rest
```

This parses the input string into four parameters as follows:

```
p1: First parameter
p2: Second parameter
p3: Third parameter
Rest: The rest of the parameter string
```

Another example of PARSE ARG is to ignore the remaining text by using a period (.) at the end of the statement. For example:

```
parse arg p1 p2 p3 p4 .
```

MSGVAR(n) does not display a fifth input parameter because it was not coded. An alternative implementation is using the PARMCNT() function to determine the number of parameters with a loop similar to the following text:

```
Do i = 1 to PARMCNT()
 Say 'Parameter number' i 'is:' MSGVAR(i)
End
```

By default, the input parameters convert to upper case. You can control the result by using NETVASIS or by defining the EXEC as a command in member CNMCMDU and coding the FOLDUP=Y definition:

```
CMDDEF.MYVARS.MOD=DSICCP
CMDDEF.MYVARS.FOLDUP=Y
```

### Step 5

By default, PARSE ARG is blank-delimited. When commas are used in the input, the PARSE ARG treats them as part of the text string.

NetView MSGVAR(n) parse is based on blanks, commas, single quotations, and the equal sign. MSGVAR(n) uses slightly more CPU cycles than EXEC because the EXEC leaves the REXX interpreter and launches into a NetView environment.

The following text is an example of coding MYVAR for this lab exercise:

```

/* REXX */
/* This is a simple REXX EXEC for L4 EX1 */
/* */

parse arg argstring .
If argstring <> '' then
Do

 /* Use REXX PARSE ARG: parse input and write to operator */
 parse arg p1 p2 p3 rest /* input parsed into 4 parms */

 Say 'MYVARS: Input parsed with PARSE ARG'
 Say ' Parameter 1 is ' p1
 Say ' Parameter 2 is ' p2
 Say ' Parameter 3 is ' p3
 Say ' Parameter 4 is ' rest

 /* Use NetView MSGVAR(n): parse input, write to operator */
 Say 'MYVARS: Input parsed with NetView MSGVAR(n) '
 Say ' Parameter 1 is ' MSGVAR(1)
 Say ' Parameter 2 is ' MSGVAR(2)
 Say ' Parameter 3 is ' MSGVAR(3)
 Say ' Parameter 4 is ' MSGVAR(4)

End

/* Use NetView built-in functions, write to operator */

Say 'Current sysplex name:' Sysplex()
Say 'Current system name :' CurSys()
Say 'Current z/OS level:' MVSlevel()
Say 'NetView level:' NetView('T')
Say 'Towers enabled:' Tower('*')
Say 'There are ' Words(NetView('T')) ' towers enabled.'
Say 'The third tower is:' Word(Tower('*'),3)
Say 'NetView domain ID:' Domain()
Say 'NetView operator ID:' Opid()
Say 'NetView task type:' Task()
Say 'Current z/OS console name:' Curconid()
Say 'Current VTAM LU name:' Lu()
Say 'Current VTAM Network ID:' Netid()

/* Use REXX provided functions and write to operator */
Say 'Current date is:' Date('E')
Say 'Current time is:' Time()

Day_Name = Date('W')

Say "Today's day is:" Day_Name

/* Use PARSE PULL to pause for operator input */

```

```

Resp = ''
Do While Resp = '' /* Loop: Resp Y or N */
 Say 'Is the day correct? Enter "GO Y" or "GO N"'
 parse pull resp
 If Resp = 'Y' Then
 Say 'You responded with a:' resp
 Else
 Do
 Say 'Incorrect response.'
 Resp = ''
 End
End /* Loop: Resp Y or N */

Say 'End of MYVARS lab exercise'

Exit

```

## Exercise 1-2: Global variables

### Lab exercise objectives

This lab provides an introduction to REXX and global variables for NetView. At the end of this lab, you should be able to perform the following tasks:

- Use the QRYGLOBL command.
- Retrieve, update, and store common global variables.

### Lab exercise instructions

- \_\_\_ 1. Access your system and log on to TSO. Follow the instructions that the instructor provides. Log on to NetView AOFDA domain, also following the instructions.
- \_\_\_ 2. From AOFDA, issue **HELP QRYGLOBL** to display the online help for the QRYGLOBL command. Review the information about displaying global variables.

What command do you issue to display the common global variables that begin with CNMSTYLE.AUTO? \_\_\_\_\_

3. Issue the QRYGLOBL command by using the NetView WINDOW function. Your result should look similar to the following text:

```
BNH031I NETVIEW GLOBAL VARIABLE INFORMATION
BNH103I COMMAND ISSUED AT: 06/15/11 10:13:23
BNH061I
BNH032I COMMON GLOBAL VARIABLES
BNH036I GLOBAL VARIABLE NAME: GLOBAL VARIABLE VALUE:
BNH061I -----
BNH039I CNMSTYLE.AUTO.MVSCMDREVISION MVSCMDS
BNH039I CNMSTYLE.AUTO.SAVEDBMAINT DBAUTO1
BNH039I CNMSTYLE.AUTO.EMAAUTOM AUTO1
BNH039I CNMSTYLE.AUTO.AUTOIP AUTOAON
BNH039I CNMSTYLE.AUTO.PKTS.TCPIP AUTOPKTS
BNH039I CNMSTYLE.AUTO.HMONDBMAINT DBAUTO2
BNH039I CNMSTYLE.AUTO.TCPSM.TCPIP AUTOTCPS
BNH039I CNMSTYLE.AUTO.NAPOLTSK2 AUTODC2
BNH039I CNMSTYLE.AUTO.NAPOLTSK3 AUTODC3
BNH039I CNMSTYLE.AUTO.NAPOLTSK1 AUTODC1
BNH039I CNMSTYLE.AUTO.NAPOLTSK4 AUTODC4
BNH039I CNMSTYLE.AUTO.SMONDBMAINT DBAUTO1
BNH039I CNMSTYLE.AUTO.OPKT.TCPIP AUTOOPKT
BNH039I CNMSTYLE.AUTO.COLTSK5 AUTOCT5
BNH039I CNMSTYLE.AUTO.XCFDISC AUTOXDSC
BNH039I CNMSTYLE.AUTO.TCPCDBMAINT DBAUTO2
BNH039I CNMSTYLE.AUTO.MEMSTORE AUTO2
BNH039I CNMSTYLE.AUTO.ENTDATA AUTOEDAT
BNH039I CNMSTYLE.AUTO.NVSOAPTSK AUTONVSP
BNH039I CNMSTYLE.AUTO.IDLEOFF AUTO1
BNH039I CNMSTYLE.AUTO.XCF AUTOXCF
BNH039I CNMSTYLE.AUTO.COLTSK7 AUTOCT7
BNH039I CNMSTYLE.AUTO.APSERV AUTOTMSI
BNH039I CNMSTYLE.AUTO.MASTER AUTO1
BNH039I CNMSTYLE.AUTO.NALCLOP AUTONALC
BNH039I CNMSTYLE.AUTO.POLICY AUTOAON
BNH039I CNMSTYLE.AUTO.TCPCONN.TCPIP AUTOTCPC
BNH039I CNMSTYLE.AUTO.NETCONV AUTO2
BNH039I CNMSTYLE.AUTO.DLAAUTO AUTO2
BNH039I CNMSTYLE.AUTO.COLTSK6 AUTOCT6
BNH039I CNMSTYLE.AUTO.PRIMARY AUTO1
BNH039I CNMSTYLE.AUTO.FKXPKTS AUTOPSAV
BNH035I NUMBER OF VARIABLES FOUND: 32
BNH061I
BNH033I TASK GLOBAL VARIABLES FOR NETOP1
BNH036I GLOBAL VARIABLE NAME: GLOBAL VARIABLE VALUE:
BNH061I -----
BNH035I NUMBER OF VARIABLES FOUND: 0
BNH061I

NETVIEW GLOBAL VARIABLE INFORMATION COMPLETE
```

These are all of the automation operators (autotasks) that are defined to NetView in CNMSTYLE. For example, XCF processing is scheduled under the CNMSTYLE.AUTO.XCF task (AUTOXCF).



**Tip:** During the lab exercises, you can test to see if a GLOBALV PUTx command works by using the QRYGLOBL command to retrieve the global variable value.

- \_\_\_ 4. Issue **HELP GLOBALV** to display the online help for the GLOBALV command. The *Help GLOBALV* panel (EUYSLIST) should open. Select options as follows:
  - A: REXX GLOBALV
  - C: GLOBALV DEF
  - D: GLOBALV GET
  - F: GLOBALV PUT
- \_\_\_ 5. Create a new REXX EXEC named MYGLOB as follows:
  - \_\_\_ a. Use PARSE ARG or MSGVAR(*n*) to parse the input string into the three following parameters:
    - Global function type: DEFT, DEFC, GETT, GETC, PUTT, or PUTC
    - Global variable name
    - Global variable value
  - \_\_\_ b. Issue the appropriate GLOBALV command to define, retrieve, or put the value for the global variable (parameter two).



**Tip:** Use the REXX **INTERPRET** instruction before your GLOBALV PUTx command, such as the following text example:

```
Interpret VarName '=' VarValue
```

In this case, the INTERPRET instruction sets the second parameter (VarName) to the value of the third parameter (VarValue).

- \_\_\_ c. Test the return code from the GLOBALV command and display a message similar to the following text:
 

```
GLOBALV Return code: Rc
```
- \_\_\_ d. For GETx requests, also display the value of the global variable after checking the return code.



**Tip:** Use the REXX **Value()** function to display the value of the global variable, such as the example text:

```
If Substr(ReqType,1,3) = 'GET' then
 Say 'Value of variable ' VarName ' = ' Value(VarName)
```

This displays the variable (VarName) and its value using Value(VarName).

- \_\_\_ e. When finished, save your work.
- \_\_\_ 6. From AOFDA, issue **MYGLOB GET MYVAR**. What response did you get?  
\_\_\_\_\_
- \_\_\_ 7. Correct the mistake and reissue the **MYGLOB EXEC**. What response did you get? Why?  
\_\_\_\_\_
- \_\_\_ 8. Issue **MYGLOB GETT MYVAR.\***. What does the return code mean?  
\_\_\_\_\_
- \_\_\_ 9. Issue **MYGLOB PUTT MYVAR\_COUNT 99**, followed by **MYGLOB GETT MYVAR\_COUNT**. Your response should be similar to the following text:  
Value of variable MYVAR\_COUNT = 99
- \_\_\_ 10. In TSO, create another REXX EXEC, naming it **OPERGLOB**. **OPERGLOB** is to issue **GLOBALV PUTx** and **GETx** commands to set several global variables for the following items:
  - Your name
  - Timestamp: Date and time that **OPERGLOB** was last issued.
  - Counter: Number of times that **OPERGLOB** has been called.
  - Update of the date and time as well as the counter every time the EXEC is called.

These variables need to be saved after you log off. What type of variable do you need to use? \_\_\_\_\_

When **OPERGLOB** runs, write messages to the operator similar to the following text:

```
My Name is ... Lab User 1
OPERGLOB was last used ... 06/15/11 06:33:59.469681
OPERGLOB was called 3 times
```
- \_\_\_ 11. Run the **OPERGLOB EXEC** several times and note the timestamp as well as the counter. Log off from AOFDA.

- \_\_\_ 12. Log on to AOFDA as the same operator ID and issue the OPERGLOB EXEC. The timestamp and counter within the output should update.

```
My Name is ... Lab User 1
OPERGLOB was last used ... 06/15/11 06:34:12.623707
OPERGLOB was called 4 times
```

If update does not occur, correct the problem in OPERGLOB and repeat these steps again.

Explain how you would reset these variable values.

---

---

---



## Answers to lab exercise 2 questions

### Step 2

Query the common global variables set by CNMSTYLE for autotasks:

```
QRYGLOBL COMMON VARS=CNMSTYLE.AUTO*
```

### Step 6

GLOBALV yields a return code 52 because the second parameter was misspelled: GET instead of GETT.

### Step 7

GLOBALV yields a return code 0 because the command syntax was correct. The variable value would be null because it had not been set with a GLOBALV PUTT command.

### Step 8

GLOBALV yields a return code of 14004 when the variable name is not valid.

### Step 10

Because this information is needed after you log off, you use common global variables.

### Step 12

Reset the value of each common global variable by setting the value of each variable to null and issuing a GLOBALV PUTC for all three variables.:

# Exercise 1-3: Global variables tracing

## Lab exercise objectives

This lab provides an introduction to REXX and global variables tracing and automation. At the end of this lab, you should be able to set up tracing and automation for global variables.

## Lab exercise instructions

- \_\_\_ 1. Access your system and log on to TSO. Follow the instructions that the instructor provides. Log on to NetView AOFDA domain, also following the instructions.
- \_\_\_ 2. Create a REXX EXEC called MYGLOB2 to perform the following tasks:
  - \_\_\_ a. Write common global variable KVAR1 with the initial value of 'AUTOMATION START'.
  - \_\_\_ b. Write common global variable KVAR2 with initial value of 'TRACE START'.
  - \_\_\_ c. Automate common global variable KVAR1.
  - \_\_\_ d. Trace common global variable KVAR2.
  - \_\_\_ e. Change the value of common global variable KVAR1 to 'AUTOMATION CHANGED' and write it.
  - \_\_\_ f. Change the value of common global variable KVAR2 to 'TRACE CHANGED' and write it.
- \_\_\_ 3. Run MYGLOB2 and view the NetView log. Entries that resulted from tracing and automating the common global variables should be similar to the following text:

```
DWO994I Common global variable KVAR1 set to value ---
>AUTOMATION CHANGED
DWO990I Common global variable 'KVAR2' set by 'GLOBALV' via
'MYGLOB2--->GLOBALV' to value --->TRACE CHANGED<---
```



**Note:** By default, the DWO994I message is not displayed or logged. We have defined an automation table entry to log the message:

```
IF MSGID = 'DWO994I' THEN NETLOG(Y);
```

The difference between Trace and Automate is information detail. Automate informs that the global variable is changed and old and new values. Trace provides more information about the change to the global variable. Both DWO990I and DWO994I messages can be automated.

## Answers to lab exercise 3 questions

```

/* REXX */
/* MYGLOB2 */
say '*****'
say '* Now we set KVAR1 and KVAR2'
say '*****'
KVAR1='AUTOMATION START'
'GLOBALV PUTC KVAR1'
KVAR2='TRACE START'
'GLOBALV PUTC KVAR2'
say '*****'
say '* Now we automate and trace common variables KVAR1 KVAR2'
say '*****'
'GLOBALV AUTO ON * KVAR1'
'GLOBALV TRACE ON * KVAR2'
say '*****'
say '* Now we change KVAR1 and KVAR2'
say '*****'
KVAR1='AUTOMATION CHANGED'
'GLOBALV PUTC KVAR1'
KVAR2='TRACE CHANGED'
'GLOBALV PUTC KVAR2'

```

## Exercise 1-4: TRAP and WAIT

This lab provides an introduction to programming REXX EXECs for NetView that issue commands and trap their responses. For example, a REXX EXEC can be driven from the automation table to take actions based on notification of a resource failure. The REXX EXEC might need to issue commands to collect more data about the resource or the failure event before taking corrective actions.

### Lab exercise objectives

At the end of this lab, you should be able to:

- Identify messages to be trapped.
- Code an EXEC to wait for one or more trapped messages. (You will also have to continue waiting for messages within your REXX EXEC.)
- Read a trapped message.
- Parse the response to a command that is issued to generate one or more trapped messages.
- Code the appropriate automation table statements to drive a user REXX EXEC.

### Lab exercise instructions

Create a REXX EXEC to automate the archival and switching of the NetView DSILOG VSAM data set. Also code automation table statements to call your EXEC.

When DSILOG switches from the primary VSAM data set to the secondary VSAM data set, the following messages occur:

```
DSI556I DSILOG : VSAM DATASET 'OPEN' COMPLETED, DDNAME = 'DSILOGS'
RETURN CODE = X'00', ACB ERROR FIELD = X'00'
DSI547I DSILOG : SECONDARY VSAM DATA SET IS NOW ACTIVE
DWO520I DSILOG : VSAM DATASET 'CLOSE' COMPLETED, DDNAME =
'DSILOGP' RETURN CODE = X'00', ACB ERROR FIELD = X'00'
```

You see similar messages when the switch occurs from secondary to primary with a DSI546I issued instead of DSI547I:

```
DSI546I DSILOG : PRIMARY VSAM DATA SET IS NOW ACTIVE
```

- \_\_\_ 1. Create a new REXX EXEC, LOGAUTO, to perform the following steps:
  - \_\_\_ a. Parse a single input parameter to the EXEC: PRIMARY or SECONDARY.
  - \_\_\_ b. Issue the LOGMAINT (user-written command that the instructor provides) EXEC to perform database maintenance on the inactive log file.

- \_\_\_ c. Test the return code when running LOGMAINT.
  - If the return is zero, continue running LOGMAINT.
  - If the return code is non-zero, issue an error message.
- \_\_\_ d. When driven for a switch to secondary (DSI547I), define a list of messages to TRAP for a NetView SWITCH command:
  - Trap and display the DSI547I message indicating a switch from primary to secondary.
  - Trap and suppress the DSI556I and DWO520I messages.
- \_\_\_ e. When driven for a switch from secondary to primary, do not issue any further commands. This archives the contents of the secondary log data set and waits for the primary to fill before it automatically switches.
- \_\_\_ f. WAIT for the messages to be trapped. Use the WAITTIME common global for the amount of time to wait, in seconds.
- \_\_\_ g. Code for each of the possible event types as follows:
  - **M**: Process the messages from the trap message queue.
  - **T**: Issue a message indicating a timeout occurred.
  - **E**: Issue a message indicating an error occurred.
  - **G**: Issue a message that the operator entered the **GO** command.



**Tip:** Refer to the lecture slide, “Example: Trapping a single message”, for an example of the REXX coding that this lab exercise requires.

You can use REXX PARSE or NetView functions, such as MSGID(), MSGSTR(), and MSGVAR(*n*) to parse the messages that are received.

- \_\_\_ h. Make sure you code the **FLUSHQ** and **TRAP NO MESSAGES** commands.
- \_\_\_ 2. Create a new automation table, LOGTBL, to process the DSI546I and DSI547I messages:
  - \_\_\_ a. Process only the messages for task name of DSILOG.
  - \_\_\_ b. Pass the log data set type (PRIMARY or SECONDARY) to the LOGAUTO EXEC.  
See the lab exercise answers for an example automation table.
- \_\_\_ 3. Issue a **LIST DSILOG** command. If the primary data set is not active, issue a **SWITCH DSILOG,P** command to switch logging to the primary data set before you test your EXEC and automation table.
- \_\_\_ 4. Issue **AUTOTBL MEMBER=LOGTBL,TEST** to test the syntax of your automation table. Correct errors before attempting to activate the automation table.
- \_\_\_ 5. Use **AUTOMAN** to insert the new automation table after DSITBL01.

- \_\_\_ 6. Issue a **SWITCH DSILOG,S** command. The automation statements you coded in LOGTBL should perform the following steps:
  - \_\_\_ a. Drive LOGAUTO to call LOGMAINT for the primary data set.
  - \_\_\_ b. Automatically issue a SWITCH DSILOG,P command.

LOGAUTO should be driven a second time to perform database maintenance on the secondary data set.



**Note:** Each time LOGAUTO runs, you should notice the *wait indicator (W)* on your NetView screen. The *wait indicator* flickers as each new trapped message processes.

You should see the following messages:

```
DSI556I DSILOG : VSAM DATASET 'OPEN' COMPLETED, DDNAME = 'DSILOGS'
RETURN CODE = X'00', ACB ERROR FIELD = X'00'
```

```
DSI547I DSILOG : SECONDARY VSAM DATA SET IS NOW ACTIVE .
```

```
DWO520I DSILOG : VSAM DATASET 'CLOSE' COMPLETED, DDNAME =
'DSILOGP'RETURN CODE = X'00', ACB ERROR FIELD = X'00'
```

The DSI556I, DSI547I, and DWO520I messages result from the SWITCH command you just entered. Subsequent DSI556I and DWO520I should be suppressed by the LOGAUTO EXEC.

```
++++ LOGMAINT: Performing database maintenance on PRIMARY
DSI546I DSILOG : PRIMARY VSAM DATA SET IS NOW ACTIVE
++++ LOGMAINT: Performing database maintenance on SECONDARY
```

The LOGMAINT EXEC issues an informational message.

The DSI546I and DSI547I messages should be held (alternately) on your screen until the log data set switches, based on statements supplied in DSITBL01. If your results differ, modify your LOGAUTO EXEC and issue the **SWITCH DSILOG,S** command again to test your changes.

- \_\_\_ 7. When you finish, use **AUTOMAN** to unload the LOGTBL automation table.

## Answers to lab exercise 4 questions

### Step 1

Your automation table should look similar to the following text:

```
if msgid='DSI546I' | msgid='DSI547I' & token(2) = 'DSILOG'
 & token(4) = PriOrSec then
 EXEC(CMD('LOGAUTO ' PriOrSec) ROUTE(ONE *)) HOLD(Y);
```

## Exercise 1-5: EXECIO

### Lab exercise objectives

This lab is optional. This lab provides an introduction to programming REXX EXECs for NetView that read members of a data set. At the end of this lab, you should be able to perform the following tasks:

- Use TRAP, WAIT and MSGREAD commands, along with MSGID(), MSGSTR(), and MSGVAR() functions.
- Allocate a data set and member to NetView.
- Use EXECIO to read the member from the data set.
- Deallocate the data set and member from NetView.



**Note:** Your instructor might provide a different data set and member to use for this exercise:

NV390.V6R1M0.USER.DSIPARM(RMTSTGEN)

---

### Lab exercise instructions

- \_\_\_ 1. Create a new EXEC, READMEM, to read member RMTSTGEN to perform the following steps:
  - \_\_\_ a. Parse the data set name and member as a single input parameter.
  - \_\_\_ b. Test the input and issue an error message if it is null.
  - \_\_\_ c. Optionally, test the input to make sure it contains a data set name and member name. Use NV390.V6R1M0.USER.DSIPARM(RMTSTGEN).
  - \_\_\_ d. Use the NetView ALLOCATE command to allocate the data set as shared (SHR) and free it when complete (FREE).



**Tip:** Use the online help for the ALLOCATE command.

If the ALLOCATE command is successful, a CNM272I message is displayed as follows:

```
CNM272I ddname IS NOW [ALLOCATED | DEALLOCATED]
```

You need to trap the CNM272I and parse the *ddname* so that EXECIO can read from the *ddname*.



- \_\_\_ e. Use EXECIO to read the contents of the data set member (identified by *ddname*) into a REXX stem variable.
- \_\_\_ f. Loop through the REXX stem variable, issuing a SAY command for each line.
- \_\_\_ g. When the loop ends (finished displaying the contents of the stem variable), issue a message identifying that the read has completed. Use EXECIO to close the file.
- \_\_\_ 2. When finished, save your work.
- \_\_\_ 3. In NetView AOFDA domain, issue **READMEM NV390.V6R1M0.USER.DSIPARM(RMTSTGEN)**. When successful, the output should be displayed similar to the following text:

```
File NV390.V6R1M0.USER.DSIPARM(RMTSTGEN)/SYS00275 has 8 lines:
***** ONLY USED FOR EXECIO LAB *****
RMTINIT.PORT = 4022
RMTINIT.IP = Yes
RMTINIT.TCPNAME = &CNMTCPN
RMTINIT.SOCKETS = 50 //
RMTINIT.KEEPALIVE = 10 //
RMTSYN.&CNMNETID..AOFDB = 10.31.&IPBSUB..&IPBLLQ.
RMTSYN.&CNMNETID..AOFDA = 10.31.&IPASUB..&IPALLQ.
READMEM EXEC is now finished
```





# **IBM Tivoli NetView for z/OS 6.1: NetView PIPEs**

## **Student Exercises**

Course: TZ233 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](http://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](http://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

|||||

|                                                                   |      |
|-------------------------------------------------------------------|------|
| Lab exercise overview                                             | 1-1  |
| Data sets used in this lab                                        | 1-1  |
| Debugging errors                                                  | 1-1  |
| Exercise 1-1: Sending messages to NetView console with PIPE       | 1-2  |
| Lab exercise instructions                                         | 1-2  |
| Answers to pipelines lab exercise 1 questions                     | 1-4  |
| Exercise 1-2: Issuing commands within a pipeline                  | 1-5  |
| Lab exercise instructions                                         | 1-5  |
| Answers to pipelines lab exercise 2 questions                     | 1-7  |
| Exercise 1-3: Reading from a data set                             | 1-8  |
| Lab exercise instructions                                         | 1-8  |
| Answers to pipelines lab exercise 3 questions                     | 1-9  |
| Exercise 1-4: Determining EXECs loaded in storage                 | 1-10 |
| Lab exercise instructions                                         | 1-10 |
| Answers to pipelines lab exercise 4 questions                     | 1-12 |
| Exercise 1-5: System symbolic variables                           | 1-13 |
| Lab exercise instructions                                         | 1-13 |
| Answers to pipelines lab exercise 5 questions                     | 1-14 |
| Exercise 1-6: More PIPE stages                                    | 1-15 |
| Lab exercise instructions                                         | 1-15 |
| Answers to pipelines lab exercise 6 questions                     | 1-17 |
| Exercise 1-7: Displaying all active address spaces in the sysplex | 1-18 |
| Lab exercise instructions                                         | 1-18 |
| Answers to pipelines lab exercise 7 questions                     | 1-20 |
| Exercise 1-8: EDIT stage                                          | 1-21 |
| Lab exercise instructions                                         | 1-21 |
| Answers to pipelines lab exercise 8 questions                     | 1-23 |
| Exercise 1-9: Complex pipeline with REXX                          | 1-24 |
| Lab exercise instructions                                         | 1-24 |
| Answers to pipelines lab exercise 9 questions                     | 1-25 |



# Student exercises for Unit 1

.....

## Lab exercise overview

In this exercise, you log on to NetView® and access the online help that is available for the PIPE command and stages. You issue several basic PIPE commands from the NetView command line, and code REXX EXECs to issue PIPE commands.

You should have two PCOMM sessions for this lab exercise. One session is for NetView and the other is for TSO.

During the lab exercises, you use several PIPE stages. If needed, use your class materials and the online help to assist you with completing the lab exercises. Examples are as follows:

- **HELP PIPE SYNTAX:** Displays help for the PIPE command and its syntax.
- **HELP PIPE *stage\_name*:** Displays help for the specified PIPE stage.

## Data sets used in this lab

NV390.WORKSHOP.CNMCLST\_\_\_\_\_

## Debugging errors

When coding a pipeline, you might not see the expected results of a PIPE stage. Insert a CONSOLE stage before or after the stage to compare the input stream versus the output stream and the effect of your PIPE stage.

# Exercise 1-1: Sending messages to NetView console with PIPE

## Lab exercise instructions

- \_\_\_ 1. Access your system and log on to NetView, following the instructions your instructor provides you.
- \_\_\_ 2. During this lab exercise, issue the NetView PIPE command from the command line. Several of the PIPE commands require more than one line of command input. Expand the number of lines for command input from one line (default) to three lines by typing the **INPUT 3** command.



**Note:** During these exercises, you can use online help for the PIPE command syntax and stages.

- \_\_\_ 3. On the command line, create a PIPE command that writes a literal character string to the NetView screen. Use the LITERAL stage to insert the character string *This is a message from TEAMx* (where *x* is your team number that the instructor assigned). Insert the string into the pipeline and the CONSOLE stage to display the string at your NetView screen.  
What does your PIPE command look like?

---

- \_\_\_ 4. On the command line, create a PIPE command that accomplishes the following tasks:
  - \_\_\_ a. Writes three mixed-case literal strings to your NetView screen.
  - \_\_\_ b. Clears the screen before writing the contents of the pipeline to the screen.
  - \_\_\_ c. Does not log the data.Use the three following literal strings:
  - THIS IS LINE 1
  - ThIs Is LiNe 2
  - this is line 3



**Note:** You should use three LITERAL stages and the CONSOLE stage.

What does your PIPE command look like?

---

What was the order of output lines? \_\_\_\_\_



- \_\_\_ 5. Modify the PIPE command and insert a COLLECT stage before the CONSOLE stage.

What effect did that have on the output? \_\_\_\_\_

\_\_\_\_\_

- \_\_\_ 6. Test the CORRWAIT stage. Issue a PIPE with stages tasks as follows:

- Insert literal string LET’S WAIT FOR 13 SECONDS.
- Wait for 13 seconds.
- Append a second literal string, WAIT ENDED.
- Display the output on the NetView screen.

Describe the output and when it occurs on the screen: \_\_\_\_\_

\_\_\_\_\_

## Answers to pipelines lab exercise 1 questions

### Step 3

The PIPE command should look like the following text:

```
PIPE LIT /THIS IS A MESSAGE FROM TEAMx/ | CONSOLE
```

### Step 4

The PIPE command should look like the following text:

```
NETVASIS pipe lit /THIS IS LINE 1/ | LIT /ThIs Is LiNe 2/ | LIT /
this is line 3/ | CONSOLE ONLY CLEAR
```

The order of output is *last in first out* (LIFO):

```
| AOFDA this is line 3
| AOFDA ThIs Is LiNe 2
| AOFDA THIS IS LINE 1
```

You can change the order of the literal strings as follows:

- Include a REVERSE STREAM stage:

```
NETVASIS pipe lit /THIS IS LINE 1/ | LIT /ThIs Is LiNe 2/ | LIT /
this is line 3/ | reverse stream | cons only clear
```

- Or use the APPEND stage:

```
NETVASIS pipe lit /THIS IS LINE 1/ | APPEND LIT /ThIs Is LiNe 2/ |
APPEND LIT /this is line 3/ | cons only clear
```

Notice the use of **NETVASIS** to maintain the case of each literal string.

### Step 5

Adding the COLLECT stage combines the three single-line messages into one multiline message:

```
| AOFDA
this is line 3
ThIs Is LiNe 2
THIS IS LINE 1
```

### Step 6

You should see the following text:

LET'S WAIT FOR 13 SECONDS when the PIPE starts.

WAIT ENDED when the PIPE ends, 13 seconds later.

Your PIPE command should look similar to the following text:

```
pipe lit /let's wait for 13 seconds/ | wait 13 | append lit /WAIT
ENDED/ | CONS
```

## Exercise 1-2: Issuing commands within a pipeline

A common use of the NetView PIPE command is issuing a command within the pipeline and passing the output of the command. The output goes to one or more pipeline stages for modifying the data in the pipeline.

Only commands that produce correlated output can be used in a pipeline. Such examples are MVS commands, VTAM commands, and most NetView commands. Also, some commands (for example, MVS) produce asynchronous output and require additional PIPE stages.

### Lab exercise instructions

1. Use the NETVIEW stage to issue a **LIST \*** command and display the output at your NetView screen in reverse video blue.



**Tip:** You should use the NETVIEW, COLOR, and CONSOLE stages.

What does your PIPE command look like?

\_\_\_\_\_

2. Use the MVS stage to issue an **MVS D T** command and the CONSOLE stage to display the response on your NetView screen.

Your command should look similar to: **PIPE MVS D T | CONS**

What output do you see? \_\_\_\_\_

Why? \_\_\_\_\_

3. To see asynchronous output from commands, use the CORRWAIT or CORRCMD stages. Add the CORRWAIT stage with a time of 10 seconds and re-issue the PIPE command.

Your command should look similar to: **PIPE MVS D T | CORR 10 | CONS**

You should see the complete response to the MVS D T command. The pipeline waits the full 10 seconds for responses.

4. Add the TAKE FIRST 1 stage to create a *terminating condition* for the CORRWAIT stage. This stage takes the first message received, ending the wait state.

Your command should look similar to: **PIPE MVS D T | CORR 10 | TAKE FIRST 1 | CONS**

You should see the complete response to the MVS D T command. The pipeline ends when the IEE136I message is received because of the TAKE stage.

- \_\_\_ 5. Use the CORRCMD stage to produce correlated output to the MVS D T command.

Your command should look similar to: **PIPE CORRCMD MVS D T | CONS**

The CORRCMD stage inserts a CORWAIT 1 stage into the pipeline that is based on the definitions in DSICCDEF. **Browse DSICCDEF** and look for the MVS command. You should see CORWAIT 3 10 0 defined. This timeout specifies as follows:

- \_\_\_ a. 3 seconds until the first response.
- \_\_\_ b. one second (1/10th of a second) after the first single-line response.
- \_\_\_ c. No time after receipt of a multi-line message.

For messages issued according to MVS standards, this timeout provides ample time without unnecessary delay.

## Answers to pipelines lab exercise 2 questions

### Step 1

The PIPE command should look like the following text:

```
pipe netv list '' | color blue rev | cons
```

The output should look similar to the following screen capture:

```
NetView V6R1M0 Tivoli NetView AOFDA NETOP1 06/15/11 09:04:37
* AOFDA PIPE NETV LIST '' | COLOR BLUE REV | CONS
- AOFDA STATION: NETOP1 TERM: SC0TCP21
- AOFDA HCOPY: NOT ACTIVE PROFILE: DSIPROFB
- 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: NNNN (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: 29572 REXX Environments: 2 (1%)
- AOFDA Messages Pending: 0 Held: 0
- AOFDA WLM Service Class: Not Available
- AOFDA END OF STATUS DISPLAY
```

### Step 2

PIPE MVS D T | CONS does not display any messages because MVS commands produce *asynchronous* output. Add the CORRWAIT stage and identify a timeout value to pass data to the output stream. Also, you can add *terminating conditions* for processing messages when they are received.

## Exercise 1-3: Reading from a data set

This exercise reads CNMSTYLE and finds the statements that include other members in CNMSTYLE.

### Lab exercise instructions

Create a PIPE to read CNMSTYLE as follows:

- \_\_\_ 1. Use the read from-disk (<) stage to read the contents of CNMSTYLE and all included members into the pipeline.
- \_\_\_ 2. Use NLOCATE to ignore all comments.
- \_\_\_ 3. Use LOCATE to find all statements that contain %INCLUDE.
- \_\_\_ 4. Display the contents of the pipeline on your NetView screen in blue.
- \_\_\_ 5. If needed, use the HELP for the LOCATE and NLOCATE stages.

The output of the PIPE command should display a list of members that are included in CNMSTYLE similar to the following text:

```
%INCLUDE CNMSTPWD
%INCLUDE CNMSTASK
%INCLUDE CNMSTIDS
%INCLUDE CNMSTACT
%INCLUDE CNMSTTWR
%INCLUDE CNMSTWBM
%INCLUDE CNMSTUSR
%INCLUDE C&NV2I . STGEN
```

You should see the NetView domain ID on every line because each line is treated as a single-line message. An example PIPE command is in the answers to the lab exercises.

## Answers to pipelines lab exercise 3 questions

The PIPE command should look similar to the following text:

```
pipe < cnmstyle | nloc 1.1 /*/ | loc /%INCLUDE/ | color blue | cons
```

## Exercise 1-4: Determining EXECs loaded in storage

This exercise combines output from two sources, the MEMLIST stage and MAPCL command, into a single stem variable. You use the MEMLIST, SEP, LOCATE, CHOP, STEM, NETVIEW, and DROP stages.

### Lab exercise instructions

1. Switch to your TSO session. Create a REXX EXEC called **QRYSTOR** to find all EXECs and command lists that are loaded into storage. You can load them by using the memStore function or the LOADCL command.
  - a. In the REXX EXEC, use three PIPE commands as follows:
    - PIPE MEMLIST: Retrieve EXECs loaded with the memStore function.
    - PIPE NETV MAPCL: Retrieve the names of the EXECs and command lists that are loaded in NetView storage.
    - PIPE STEM: Display the stem variable build from each of the previous PIPE commands.
  - b. Use the MEMLIST stage to read the DSICLD data set concatenation and find only the records that are stored in data set index of zero (0).

For example, CNME1097 is loaded in storage. The output will look similar to the following text:

```
CNME1097 0
```



**Tip:** Truncate the data to keep only the EXEC name by using a **CHOP 9**.

- c. Place the output of your PIPE command into a stem variable called EXECS. Use the MAPCL \* command to locate the EXECs and command lists loaded in storage. For example, EXEC CNMETACI is loaded in storage:

```
' AOFDA
CNM429I MAPCL DISPLAY
NAME USAGE RECORDS BYTES DATE TIME DP R/C

CNMETACI 0 120 9512 06/14/11 03:13:58 -- R

1 0 120 9512 --TOTALS--
```



**Tip:** Remove all header lines and trailer lines from the pipeline. Also, truncate the data lines after the name column with a **CHOP 9**.



- \_\_\_ d. **Append** the output of your PIPE command into the EXECs. stem variable. When you have stored all data in the EXECs. stem variable, issue a PIPE command to display the contents of the stem variable on your NetView screen.
- \_\_\_ 2. Switch to your NetView session and issue the QRYSTOR EXEC. You should see output similar to the following text:

```
* AOFDA QRYSTOR
| AOFDA
CNME1505
CNME1096
CNMETACI
```



**Tip:** You can also use the WINDOW function to issue the QRYSTOR EXEC and see its output in a panel.

An example QRYSTOR EXEC is in the answers to the lab exercises.

## Answers to pipelines lab exercise 4 questions

### Step 1

The QRYSTOR EXEC should look similar to the following text:

```
/* REXX EXEC TO QUERY ALL EXECs AND CLISTS IN STORAGE */

'PIPE MEMLIST DSICLD ', /* read DSICLD concat */
' | SEP ',
' | LOC 10.1 /0/ ', /* loc in stor members */
' | CHOP 9 ', /* truncate ... */
' | STEM execs.' /* save in stem var */

'PIPE NETV MAPCL * ', /* read LOADCL members */
' | SEP ',
' | DROP FIRST 3 ', /* delete header msgs */
' | DROP LAST 2 ', /* delete trail msgs */
' | CHOP 9 ', /* truncate ... */
' | STEM execs. APPEND' /* append to stem var */

'PIPE STEM execs. | COLL | CONS'
```

## Exercise 1-5: System symbolic variables

This exercise reads CNMSTYLE and finds the statements that include other members in CNMSTYLE and resolves any symbolic variables.

### Lab exercise instructions

Retrieve your PIPE command from lab exercise 3, (Reading from a data set on page 8). Modify the PIPE to substitute symbolic variables. Your output should look similar to the following text:

```
%INCLUDE CNMSTPWD
%INCLUDE CNMSTASK
%INCLUDE CNMSTIDS
%INCLUDE CNMSTACT
%INCLUDE CNMSTTWR
%INCLUDE CNMSTWBM
%INCLUDE CNMSTUSR
%INCLUDE CNMSTGEN
```

Notice that the C&NV2I.STGEN is resolved through the use of the SUBSYM stage. The symbolic variable named &NV2I equates to two characters: NM.

## Answers to pipelines lab exercise 5 questions

The PIPE command should look similar to the following text:

```
PIPE < CNMSTYLE | NLOC 1.1 /*/ | LOC /%INCLUDE/ | COLOR BLUE |
SUBSYM | CONS
```

## Exercise 1-6: More PIPE stages

This exercise discusses the optional tasks that are defined to NetView. The tasks are defined in CNMSTYLE (and members that are included in CNMSTYLE). Task definitions in CNMSTYLE are similar to the following text:

```
TASK.DSILOG.MOD=DSIZDST // NetView's logging task
TASK.DSILOG.MEM=DSILOGBK
TASK.DSILOG.PRI=1
```



**Note:** This exercise displays the tasks that are defined to NetView but not their status.

### Lab exercise instructions

- \_\_\_ 1. Create a PIPE to perform the following tasks:
  - \_\_\_ a. Use the from-disk (<) stage to read the contents of CNMSTYLE and all included members into the pipeline.
  - \_\_\_ b. Ignore all comments.
  - \_\_\_ c. Place the **TASK** character string in position one for a length of five characters.
  - \_\_\_ d. Further refine the pipeline data by finding only those lines that contain the character string **MOD=**.
  - \_\_\_ e. All inline comments begin with two slashes (//). Delete the inline comments from the TASK. statement.



**Tip:** Use the CHOP BEFORE stage.

- \_\_\_ f. Resolve all symbolic symbols, such as the NetView domain ID.
- \_\_\_ g. Write the literal END OF TASK DEFINITIONS as the last line in the pipeline.
- \_\_\_ h. Display the contents of the pipeline on your NetView screen.

You might need to use the following PIPE stages: <, NLOCATE, LOCATE, CHOP, SUBSYM, LITERAL, REVERSE, and CONSOLE.

The output of the PIPE command should display a list of TASK. statements similar to the following text:

```
TASK.CNMTAMEL.MOD=DSIZDST
TASK.DSIACBMT.MOD=DSIACBMT
TASK.DSIWTOMT.MOD=DSIWTOMT
...
END OF TASK DEFINITIONS
```

An example PIPE command is in the answers to the lab exercises.

## Answers to pipelines lab exercise 6 questions

### Step 1

The PIPE command should look similar to the following text:

```
pipe < cnmstyle INCL | nloc 1.1 /*/ | loc 1.5 /TASK./ | loc /MOD=/
| chop before STRing ././. | SUBSYM | REVERSE STREAM | LIT /END OF
TASK DEFINITIONS/ | REVERSE STREAM | cons
```

- PIPE < CNMSTYLE INCL: Reads all lines of CNMSTYLE plus all included members and writes them to the output stream.
- NLOC 1.1 /\*/: Removes all comments.
- LOC 1.5 /TASK./: Keeps only those lines that begin with TASK. in the first five characters.

Pipeline data should look similar to the following text at this point:

```
TASK.DSILog.MOD=DSIZDST // NetView's logging task
TASK.DSILog.MEM=DSILOGBK
TASK.DSILog.PRI=1
```

- LOC /MOD=: Keeps only those lines that contain MOD= anywhere in the line.

Pipeline data should look similar to following text:

```
TASK.DSILog.MOD=DSIZDST // NetView's logging task
```

- CHOP BEFORE STRING ./.: Removes all inline commentary (beginning with //).

Comments are identified by double forward slashes (//), which can cause possible conflict with the default string delimiter of slash (/) and period (.).

- SUBSYM: Substitutes symbolic variables, including the NetView domain ID.
- REVERSE STREAM: Reverses order of messages.
- LIT /END OF TASK DEFINITIONS/: Adds the literal string into the pipeline.
- REVERSE STREAM: Reverses order of messages a second time to preserve the original order with the literal string as the last line of pipeline data.
- CONS: Displays message at the NetView console.

## Exercise 1-7: Displaying all active address spaces in the sysplex

This lab exercise requires coding in REXX to perform the following tasks:

- \_\_\_ 1. Issue commands within a pipeline.
- \_\_\_ 2. Parse the output of each command.
- \_\_\_ 3. Pass data between EXECs in SAFEs.

More complexity in a pipeline means there are several ways to accomplish a task, especially when you code the pipeline in a REXX EXEC. Your pipeline might differ from the example.

### Lab exercise instructions

Create two REXX EXECs to display all active address spaces for all systems within the sysplex:

#### EXEC1 logic:

- \_\_\_ 1. Issue the **MVS DISPLAY XCF** command to determine the systems in the sysplex. Your output should look similar to the following text:

```
IXC334I 16.00.15 DISPLAY XCF 075
 SYSPLEX PLEX12: MVSA MVSB
```

- \_\_\_ 2. Issue the **MVS DISPLAY XCF** command in a PIPE and process the response:
  - \_\_\_ a. Delete (DROP) the IXC334I line.
  - \_\_\_ b. Split the second line into four lines (one word per line).
  - \_\_\_ c. Delete (DROP) the first two lines in the pipeline.
  - \_\_\_ d. Store the remaining lines that contain the system names in a SAFE named INSAFE.
  - \_\_\_ e. Call EXEC2. (See the description of EXEC2 logic.)
  - \_\_\_ f. When EXEC2 completes, (in EXEC1) PIPE the contents of the output SAFE from EXEC2, OUTSAFE, to the NetView console as a single multiline message.

#### EXEC2 logic:

- \_\_\_ 1. Retrieve the contents of the input SAFE, INSAFE. This safe contains the system names that EXEC1 saved. The format is one system name per line of data in the SAFE.



**Tip:** Create a loop to process the system names.



- \_\_\_ 2. Use the LITERAL stage to insert a text string into the pipeline as each system is processed for identifying the name of the system when EXEC1 receives control again. An example follows:

```
'PIPE LIT /*** System Name:' system_name '/' ,
```

- \_\_\_ 3. Choose a color for the text string and make it reverse video, An example follows:

```
'| COLOR YELLOW REV ', /* Change to reverse yellow */
```

- \_\_\_ 4. Issue the **MVS ROUTE system\_name,DISPLAY A,L** command. For example, MVS ROUTE MVSB, D A,L.

- \_\_\_ 5. Delete the first three lines and the last line of the command output.

- \_\_\_ 6. Store the active applications for that system in the output SAFE, OUTSAFE. An example follows:

```
'| SAFE OUTSAFE APPEND' /* Append to safe */
```



**Tip:** Before coding your REXX EXECs, issue the MVS commands in a pipeline from the NetView command line to see what the output looks like. Use the pipelines as a prototype for your REXX PIPE stages.

PIPE stages that you might need to use are as follows: MVS, CORRWAIT (or CORRCMD), DROP, SPLIT, SEPARATE, STRIP, COLOR, LITERAL, COLLECT, SAFE, and CONSOLE. You use the SEIZE option when reading each safe to empty the safe while reading. For readability, use the NetView WINDOW function (PF10) when calling your EXEC. An example follows.

```
NetView V6R1M0 Tivoli NetView AOFDA TSCCW01 07/02/11 08:08:06
! AOFDA
*** System Name: MVSA
CANACN CANACN CNDL NSW S LLA LLA LLA NSW S
VLF VLF VLF NSW S JES2 JES2 IEFFPROC NSW S
NET NET VTAM NSW S RACF RACF RACF NSW S
RMF RMF IEFFPROC NSW S TSO TSO STEP1 OWT S
SDSF SDSF SDSF NSW S TCPIP TCPIP TCPIP NSW SO
RMFGAT RMFGAT IEFFPROC NSW SO TELNET TELNET TN3270 NSW SO
OSNMPD OSNMPD OSNMPD OWT SO SNMPQE SNMPQE SNMPQE OWT SO
FTPSEVER STEP1 STCAPP OWT AO AUTOSSI AUTOSSI NETVIEW NSW S
AUTONETV AUTONETV NETVIEW NSW SO
*** System Name: MVSB
CANACN CANACN CNDL NSW S LLA LLA LLA NSW S
VLF VLF VLF NSW S JES2 JES2 IEFFPROC NSW S
NET NET VTAM NSW S RACF RACF RACF NSW S
RMF RMF IEFFPROC NSW S TSO TSO STEP1 OWT S
SDSF SDSF SDSF NSW S TCPIP TCPIP TCPIP NSW SO
RMFGAT RMFGAT IEFFPROC NSW SO TELNET TELNET TN3270 NSW SO
OSNMPD OSNMPD OSNMPD OWT SO SNMPQE SNMPQE SNMPQE OWT SO
FTPSEVER STEP1 STCAPP OWT AO AUTOSSI AUTOSSI NETVIEW NSW S

```

Example REXX EXECs are in the answers to the lab exercises.

## **Answers to pipelines lab exercise 7 questions**

You can find a possible solution for this exercise (EXEC1 and EXEC2) in the answers that your instructor supplies.

## Exercise 1-8: EDIT stage

Command output creates pipeline data records. You can use the EDIT stage to reformat the output. This exercise shows you how to reformat the output of a LIST STATUS=TASKS command. More complexity in a pipeline means there are several ways to accomplish a task, especially when you code the pipeline in a REXX EXEC. Your pipeline might differ from the example provided.

### Lab exercise instructions

Create a REXX EXEC called EDITTEST to reformat the output of a LIST STATUS=TASKS command for optional (TYPE=OPT) tasks. Use the EDIT stage to modify the contents of the messages in the pipeline. Use the NETV, LOC, NOT CHOP, EDIT, COLOR, and CONS pipeline stages in this exercise.

- \_\_\_ 1. Begin the pipeline by issuing the LIST STATUS=TASKS command.
- \_\_\_ 2. Locate only the lines of data that contain the string OPT (optional tasks). An example line for the DSILOG task should look like the following text:  

```
TYPE: OPT TASKID: DSILOG TASKNAME: DSILOG STATUS: ACTIVE
```
- \_\_\_ 3. Use the NOT CHOP stage to truncate all text up to the string TASKNAME:. Each record in the pipeline should look like the following text:  

```
TASKNAME: DSILOG STATUS: ACTIVE
```
- \_\_\_ 4. Split the pipeline data into two streams, based on the status of the task. Create one output stream for ACTIVE tasks and the other output stream for NOT ACTIVE tasks.
  - For all ACTIVE tasks, perform the following steps:
    - \_\_\_ a. Locate all ACTIVE tasks.
    - \_\_\_ b. EDIT the pipe output so that only the task name and status (ACTIVE) pass to the output stream.

- \_\_\_ c. Display all active task messages in green.
  - For all other tasks, perform the following steps:
- \_\_\_ a. EDIT the pipe output so that only the task name and status (NOT ACTIVE) pass to the output stream.
- \_\_\_ b. Display all not-active task messages in red.

```
* AOFDA EDITTEST
- AOFDA DSIMONIT ACTIVE
- AOFDA DSITIMMT ACTIVE
- AOFDA DSIDCBMT ACTIVE
- AOFDA DSIHLLMT ACTIVE
- AOFDA DSISTMMT ACTIVE
- AOFDA DSILOGMT ACTIVE
- AOFDA DSILOG ACTIVE
- AOFDA DSITRACE NOT ACTIVE
- AOFDA DSIELTSK NOT ACTIVE
- AOFDA DSICORSV NOT ACTIVE
- AOFDA AAUTCNMI ACTIVE
- AOFDA AAUTSKLP ACTIVE
- AOFDA ALIASAPL NOT ACTIVE
- AOFDA AOFDABRW ACTIVE
- AOFDA AOFDALUC ACTIVE
- AOFDA AOFDASIR ACTIVE
- AOFDA AOFDAVMT ACTIVE
- AOFDA BNJDSERV ACTIVE
- AOFDA BNJMNPD A ACTIVE
- AOFDA CNMCALRT ACTIVE
- AOFDA CNMTAMEL ACTIVE
- AOFDA DSIACBMT ACTIVE
- AOFDA DSIAL2WS NOT ACTIVE
- AOFDA DSIAMLUT ACTIVE
- AOFDA DSIATOPT NOT ACTIVE
- AOFDA DSICRTR ACTIVE
??? ***
```

An example PIPE is in the answers to this lab exercise.

## Answers to pipelines lab exercise 8 questions

The PIPE command should look like the following text:

```
/* */

'pipe (END %) netv list status=tasks ' ,
' | loc /OPT/ ',
' | NOT CHOP 10 AFTER STRing /TASKID:/ ',
' | Active: loc /STATUS: ACTIVE/ ',
' | EDIT word 2 1 word 4 nw',
' | COLOR GRE ',
' | CONS ',
' %Active: ',
' | EDIT word 2 1 word 4 nw word 5 nw',
' | COLOR RED ',
' | CONS '
```

The *Active:* label processes all tasks with a status of ACTIVE, placing items as follows:

- Second word in the input stream (task name) to become the first word of the output stream
- Fourth word in the input stream (ACTIVE) to become the next word of the output stream
- All active tasks to be displayed in green

The remaining messages (for example, status of NOT ACTIVE) are placed into the pipeline with items as follows:

- Second word in the input stream (task name) to become the first word of the output stream
- Fourth word in the input stream (NOT) to become the next word of the output stream
- Fifth word in the input stream (ACTIVE) to become the next word of the output stream
- All inactive tasks to be displayed in red

## Exercise 1-9: Complex pipeline with REXX

This lab exercise is optional.

This exercise requires coding in REXX to issue commands within a pipeline. You must also manipulate the pipeline data in several PIPE segments through the use of END characters, labels, connectors, and the FANIN stage. More complexity in a pipeline means there are several ways to accomplish a task, especially when you code the pipeline in a REXX EXEC. Your pipeline might differ from the example provided.

### Lab exercise instructions

Create a REXX EXEC, called EXEC4, to perform the following tasks:

- \_\_\_ 1. Issue the NetView **LIST STATUS=TASKS** command in a PIPE.
- \_\_\_ 2. Process the output with multiple PIPE segments as follows:
  - List inactive OPT tasks first with color of red.
  - List active OPT tasks next with color of green.
  - List active NetView operators next with color of yellow.
  - List active automation operators (autotasks) with color of pink.
  - Do not display the NetView primary program operator interface task (PPT) or the NetView main task (MNT)
- \_\_\_ 3. Display the output on the NetView screen. Your NetView operator ID, TSCCWxx, should be in the list with a color of yellow.

An example REXX EXEC is in the answers to the lab exercises.

## Answers to pipelines lab exercise 9 questions

You can find a possible solution for this exercise (EXEC4) in the answers that your instructor supplies.

The pipeline coded in EXEC4 is as follows:

```
'WINDOW PIPE (END %) NETV LIST STATUS=TASKS ',
' | NOTOPT: LOC /OPT/ ', /* Locate type=OPT */
' | ACTOPT: LOC /NOT ACT/ ', /* Locate NOT ACTIVE */
' | COLOR RED ', /* Not active OPT are red */
' | DISP: FANIN ',
' | CON ONLY ', /* Display all lines */
' % ACTOPT: ', /* END OF ACTOPT: segment */
' | COLOR GRE ', /* Active OPT are green */
' | DISP: ',
' % NOTOPT: ', /* END OF NOTOPT: segment */
' | LOC /OST/ ', /* Locate type=OST */
' | NLOC /NOT ACT/ ', /* Remove not active OST */
' | AUTO: PICK 20.8 - 39.8', /* TASKID <> RESOURCE */
' | COLOR YEL', /* OST (oper) are yellow */
' | DISP: ',
' % AUTO: ', /* END OF AUTO: segment */
' | COLOR PINK', /* AOST (autotask) are pink */
' | DISP: '
```

