



IBM Tivoli NetView for z/OS 6.1: NetView PIPEs

Student's Training Guide

Course: TZ233 ERC: 1.0

September 2011

© Copyright IBM Corp. 2011. All Rights Reserved.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Printed in Ireland

Table of contents

Unit 1: Introduction to NetView® for z/OS PIPEs

Introduction	1-2
Objectives	1-2
Lesson 1: Overview of NetView pipelines	1-3
Analogy to a water pipeline	1-4
NetView data pipeline	1-5
NetView pipelines overview	1-6
Features of NetView pipelines	1-7
Sources of a NetView PIPE command	1-8
First stage of a PIPE	1-9
First stage of a PIPE: Examples 1 through 3	1-10
First stage of a PIPE: Examples 4 through 6	1-11
Processing PIPE data	1-12
PIPE output	1-13
Simple multiple-stage PIPE	1-14
Asynchronous command responses	1-15
TRAP and WAIT correlation issue	1-16
Correlation with PIPEs	1-17
When correlation is necessary	1-18
Correlation notes	1-19
Lesson 2: PIPE stages	1-20
Simple PIPE command syntax	1-21
Simple example of a complex pipeline	1-22
Interfacing with applications	1-23
Interacting with users	1-24
Working with files	1-25
Storing pipeline data	1-26
Filter stages (1 of 2)	1-27
Filter stages (2 of 2)	1-28
Issuing commands: NETView stage	1-29
Issuing commands: MVS stage	1-30
Simple PIPE example 1	1-31
Simple PIPE example 2	1-32
Simple PIPE example 3	1-33
MOE operand and error messages	1-34
CORRWAIT stage	1-35
CORRCMD stage	1-37
Correlation with PERSIST stage	1-38
Student exercise	1-40
Lesson 3: Using PIPEs to access files	1-41
PIPE QSAM stage	1-42
PIPE < and PIPE > stages	1-42
Security considerations	1-43
PIPE QSAM example	1-44
Example of PIPE <	1-45
MEMLIST stage example	1-46
Student exercise	1-47
Lesson 4: Pipelines and variables	1-48
VAR stage	1-49

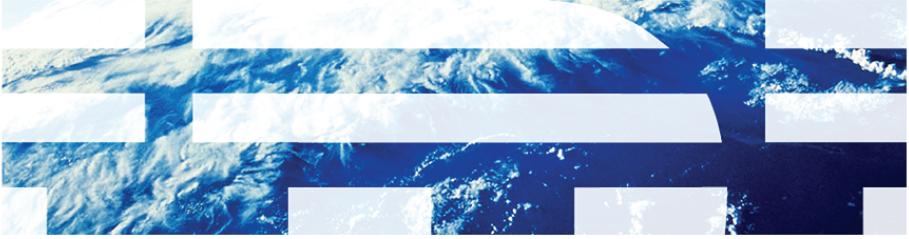
Table of contents

PIPE VAR example1-51
STEM stage1-52
Student exercise1-54
Lesson 5: Working with text in pipeline1-55
COLOR Stage: Change message attributes1-56
LOCATE stage: Selecting data from pipeline1-57
NLOCATE stage: Discarding data from pipeline1-58
TOSTRING stage: Ending data stream if string is found1-59
TAKE stage: Selecting data based on line number1-60
DROP stage: Discarding data based on line number1-61
COLLECT and SEPARATE stages: Processing messages1-62
CHOP stage: Truncating pipeline data1-63
BETWEEN stage: Dividing a message stream1-64
REVERSE stage: Reversing order of pipeline data1-65
PICK stage: Selecting specific data1-66
SUBSYM stage: Substituting symbolic variables1-67
CHANGE stage: Replacing string data in pipeline1-68
CASEI stage: comparing strings1-69
SPLIT stage: Dividing data based on string1-70
STRIP stage: Removing characters from a line1-71
SORT stage: Sorting output stream1-72
JOINCONT stage: Combining pipeline records1-73
DUPLICAT stage: Duplicating pipeline records1-74
DELDUPES stage: Removing duplicate records1-75
COUNT stage: Counting numbers of records1-76
Lesson 6: Processing pipeline data1-78
HOLE stage: Deleting all data from pipeline1-79
LOGTO stage: Log messages1-80
SAFE stage: Storing pipeline data1-81
Default SAFE1-81
Named SAFE1-82
SAFE stage examples1-83
KEEP stage1-84
KEEP example1-85
Student exercise1-86
Lesson 7: Advanced topics1-87
Cross-domain pipelines1-88
Pipe within a pipe1-89
Complex pipeline structure1-90
Complex PIPE example1-91
NOT stage: Exchanging input and output streams1-92
Multiple input and output streams1-93
FANIN, FANINANY, and FANOUT1-94
FANIN stage example1-95
Student exercise1-96
Lesson 8: PIPE EDIT stage1-97
PIPE EDIT stage example1-99
PIPE EXPOSE stage1-101
Student exercise1-103
Lesson 9: Full-screen automation1-104
Full-screen automation with a VOST1-105
Student exercise1-106
Summary1-107

Unit 1: Introduction to NetView® for z/OS PIPEs

IBM

Introduction to NetView® for z/OS PIPEs



© 2011 IBM Corp.

Introduction

This unit provides basic information of NetView® for z/OS® PIPEs and the most commonly used PIPE stages. You learn to use PIPE to perform tasks as listed in the slide.

Objectives



Objectives

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

- Discuss the basics of NetView PIPEs
- Use NetView PIPEs to perform the following tasks:
 - Issue commands
 - Trap and parse messages
 - Set and retrieve global variables
 - Read from and write to files
 - Perform automation

Lesson 1: Overview of NetView pipelines



Lesson 1: Overview of NetView pipelines

- PIPEs provides stages to perform the following tasks:
 - Issue commands
 - Process messages
 - Access files: VSAM, QSAM, DB2
 - And more
- Extends function of or reduces complexity of the following capabilities:
 - Message handling
 - Message automation, including full screen
 - Communications with non-NetView environments
- Based on UNIX pipelines

1-3

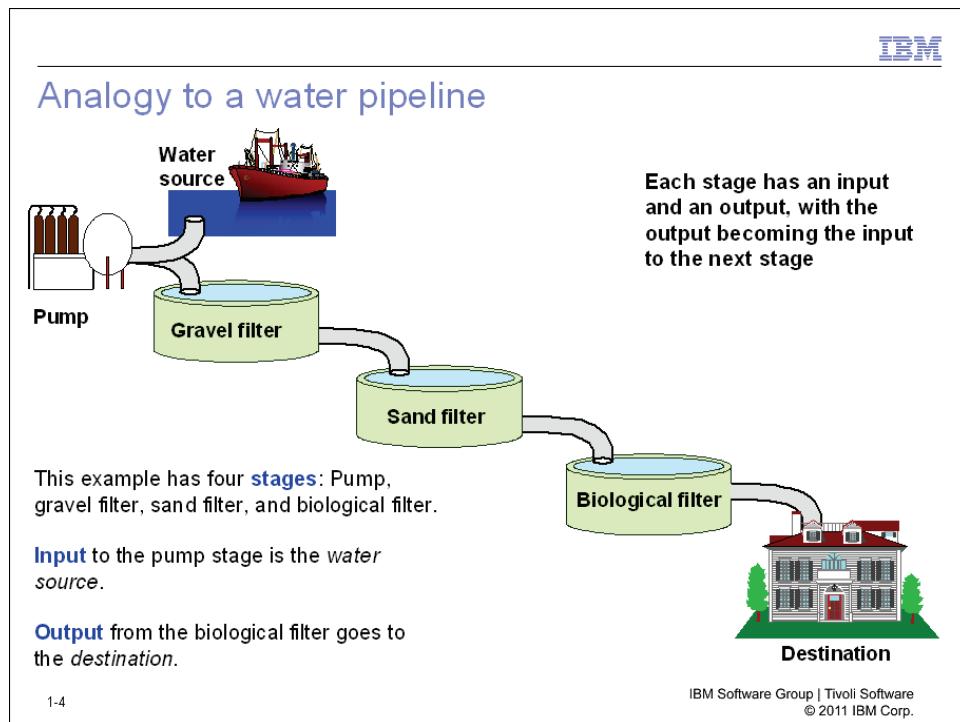
IBM Software Group | Tivoli Software
© 2011 IBM Corp.

NetView pipelines help you solve a complex problem by dividing the problem into a set of smaller, simpler steps. Each step or stage handles one part of the overall problem. PIPE stages can do the following tasks:

- Read data from several sources. For example, it can read files on disk or variables in command procedures.
- Filter and refine the data.
- Export (output) the data from the pipeline.

You can connect stages in logical sequence until they collectively cover all steps that are required to solve your problem.

Analogy to a water pipeline

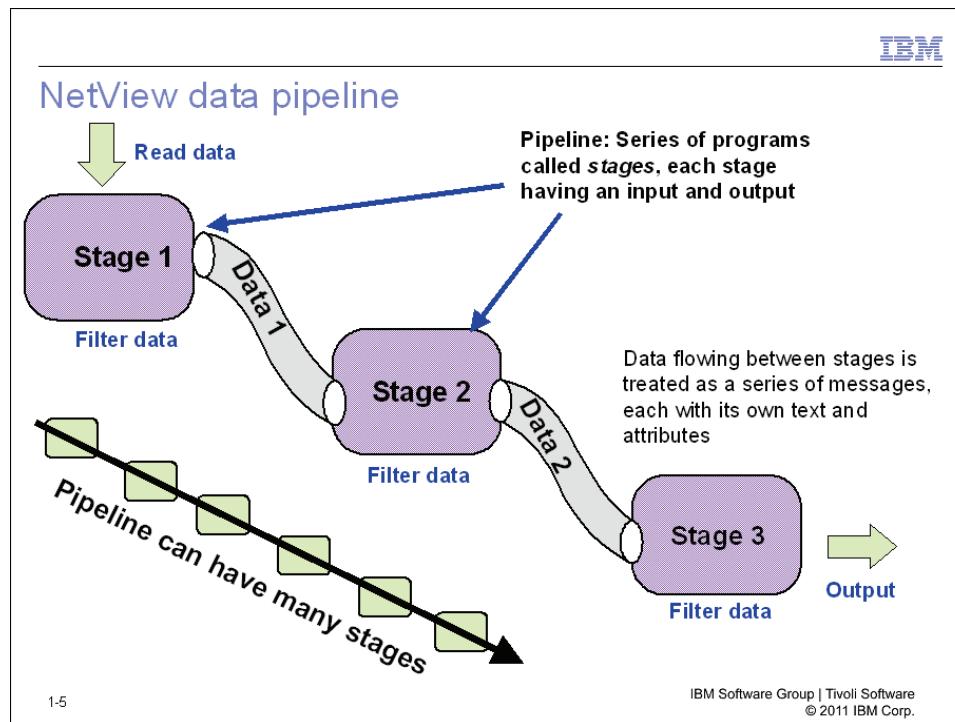


You can understand NetView pipelines if you compare them to a plumbing pipeline in a water treatment process:

Table 1: Comparing plumbing pipeline to NetView pipeline

Plumbing pipeline	NetView pipeline
Receives water from a source. An example is a reservoir or a well.	Receives data from a source. An example is a keyboard or disk.
Passes water through system.	Passes data through stages.
Combines different sizes and shapes of filters to perform complex purification processes.	Combines different stage specifications to perform complex data refinement.
Delivers purified water.	Delivers refined data to other programs or storage.

NetView data pipeline



Input data passes through a stage, and a stage command performs an action on it. Several of these stages can be linked together to form a pipeline that produces the output stream that is required.

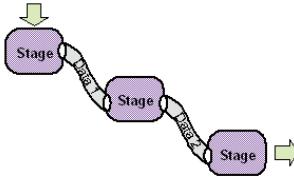
Each stage in the pipeline acts on the data in the pipeline. For example, stages can edit, delete, add, or display the pipeline data. Each stage has an *input stream* and an *output stream*. When a stage ends, its output stream becomes the input stream for the next stage.

NetView pipelines overview

IBM

NetView pipelines overview

- Series of programs: Each program is a **PIPE** stage
- Stages run within NetView **PIPE** command
- Stage separators separate stages
- Data records pass from one stage to the next:
 - Messages include both text and attributes
 - Literal strings
- Stages can modify data records
- Stages can filter data records
- Each stage can act only on data that it receives:
For example, if a previous stage removes a message, subsequent stages do not have access to that message
- You can build complex pipelines



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Data in the pipeline is a series of discrete records called messages. After it is read into the pipeline, each record becomes a message consisting of message text and message attributes. Data can be discarded, modified, split, ignored, logged, displayed, colored, and so on.

The processing is performed by *pipeline stages*. The *output stream* of one stage is the *input stream* to the next stage. Some stages support two or more input and output streams.

A multiline message is one data record. Multiline messages can be split into multiple single-line data records. Discussions about more complex pipelines (for example, nested and multi-stage pipelines) occur later in this unit.

Features of NetView pipelines



Features of NetView pipelines

- Pipelines are more efficient to code than programs
- Can improve performance
- Uses **correlated responses**, helping to prevent results as follows:
 - REXX, TRAP, and WAIT not correlated
 - Unrelated messages that might fulfill a TRAP condition, causing errors
- Various inputs and outputs
 - Files
 - Messages
 - Console
 - And so on

1-7

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Pipelines are typically more efficient to code and test than programs. Pipeline data results in correlated asynchronous responses.

Sources of a NetView PIPE command



Sources of a NetView PIPE command

REXX EXECs are most common form of pipeline

- Operators
- Automation table
- Command processors

1-8

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Many pipelines start out simple, but can quickly become complicated. Note that there is a difference between a *simple* pipeline and a *complex* pipeline. A complex pipeline has a slightly different structure than the simple type. Discussion of complex pipes occurs later in this unit.

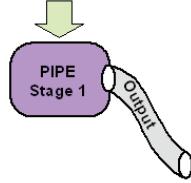
Operators can issue pipelines. REXX EXECs are the most common form of pipelines because the pipeline stages often require programming logic.

First stage of a PIPE

IBM

First stage of a PIPE

- Initial stage generates data records
- Remaining stages process the following tasks or information:
 - Command result
 - Reading records from a file
 - Reading input safe
 - Reading variables: Local or global
 - Writing a text (literal) string
 - Reading contents of a full screen panel
 - Program-to-program interface (PPI)
 - Held-message queue
 - Environment data
 - And more



The diagram shows a purple rounded rectangle labeled "PIPE Stage 1". A green arrow points down to the top of the rectangle. A curved grey arrow labeled "Output" points away from the bottom right corner of the rectangle.

1.9

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The first stage of a pipeline typically issues a command to generate data records for the remaining stages to process. This slide shows several examples of creating data for a pipeline.

First stage of a PIPE: Examples 1 through 3



First stage of a PIPE: Examples 1 through 3

- **Issue NetView command**
PIPE NETVIEW LIST STATUS=TASKS | CORR | ...
- **Issue MVS command**
PIPE MVS D T | TAKE FIRST 1 | ...
- **Read contents of NetView operator profile**
 - PIPE QSAM 'CNM.DSIPRF(DSIPROFB)' | CONS
 - PIPE < DSIPRF.DSIPROFB | CONS

1-10

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Using NetView pipelines, you can perform tasks as follows:

- Issue commands. Examples are NetView and MVS commands.
- Read data from files and write data to files. Pipeline stages are provided to support VSAM, QSAM, and DB2® or SQL files. Discussion of the QSAM and read from (<) pipe stages occur later in this unit.

First stage of a PIPE: Examples 4 through 6



First stage of a PIPE: Examples 4 through 6

- **PIPE STEM myvar. | ...**
Read contents of stem variable, myvar, into pipeline
Note: Stem index variable, Myvar.0, must be set
- **PIPE LITERAL /place this text into the pipeline/ | ...**
 - Write literal string into pipeline
 - Delimiter: First nonblank character that is encountered after LIT stage
 - Use NETVASIC for mixed case
- **pipe var (common) cnmstyle.tcpname |**
Read contents of common global variable named CNMSTYLE.TCPNAME into pipeline

1-11

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide shows examples of generating data records as the first stage of a pipeline. If LITERAL is not the first stage, messages remain in the pipeline, and text added by this stage is inserted in front of the existing messages. You can change the order of pipeline data with the REVERSE or APPEND stages.



Note: You cannot issue STEM and VAR stages from the NetView command line.

Processing PIPE data

IBM

Processing PIPE data

- First stage generates data records
- Remaining stages can perform the following tasks:
 - Locate specific data
 - Filter data
 - Split multiline data into single-line
 - Collect multiple single-lines into one multiline
 - Save data into variable
 - Sort data
 - Modify data records
 - Display or log data
 - And so on

Output from stage 1 is input to stage 2,
output from stage 2 is input to stage 3,
and so on

1-12

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The first stage in a pipeline generates data for its output stream to pass to the second stage in the pipeline. The remaining pipeline stages can locate data, sort data, modify data, and so on, passing the data in the output stream for each stage.

Pipeline stages are categorized as follows:

- Device drivers: Transport data but do not take any action on it. Device drivers write their input stream to their output stream. PIPE QSAM and PIPE NETVIEW are examples of device drivers.
- Filters: Examine their input stream, passing data that matches a specified criteria to their *primary output stream*. Data that does not match the specified criteria passes to a *secondary output stream*. PIPE LOCATE and PIPE TAKE are examples of filters.

Pipelines can handle a large amount of data without the need for all of the data to be in storage simultaneously. Some stages (for example, COLLECT) might need to handle all of the data simultaneously.

PIPE output

PIPE output

- NetView operator console
- NetView log
- System log
- Hard copy log
- PIPE SAFE and KEEP
- PIPE HOLE (to discard messages)
- REXX variable (simple or stem)
- File
- PPI
- Command execution

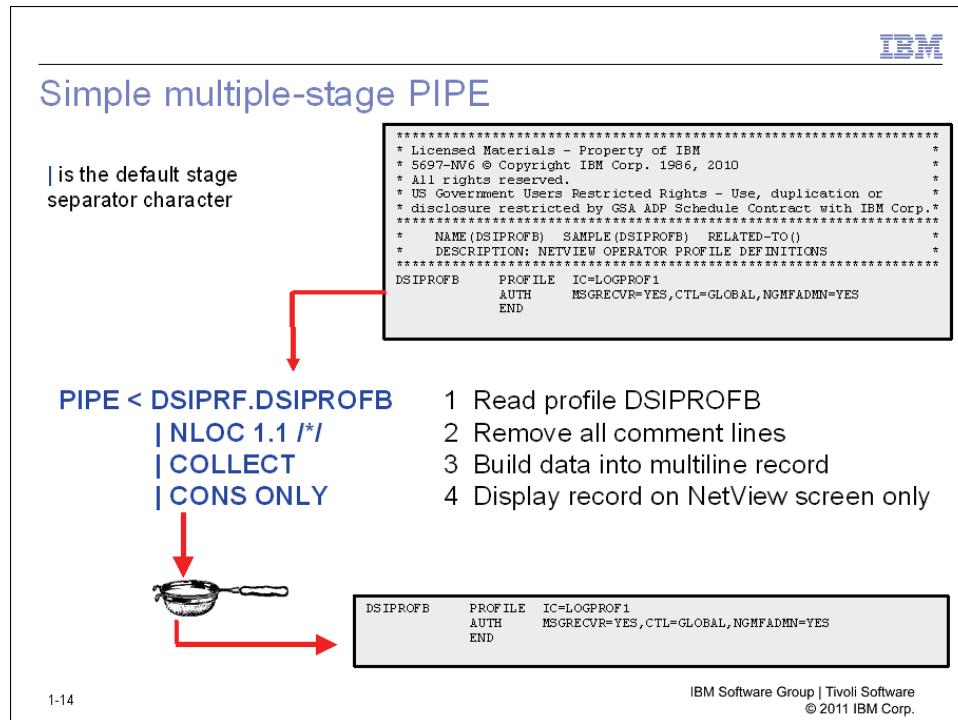
The diagram shows a purple rounded rectangle labeled "PIPE Stage". An arrow originates from the bottom right corner of this box and points to the right, labeled "Output data". Above the stage, another arrow originates from the top left corner and points upwards and to the left, labeled "Output".

1-13

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Pipeline data can be displayed, logged, discarded, written to a file, and so on. Note that the slide shows the pipeline output as the last stage. You can code any of the output stages in the middle of a pipeline. Each output stage processes the data in the input stream and passes the data (unmodified) to the output stream. For example, if you are debugging a problem within a pipeline, you can insert the CONS stage to display the contents of the pipeline.

Simple multiple-stage PIPE



This slide shows a simple example of a pipeline with four stages:

1. Read file DSIPROFB in the NetView DSIPRF data set concatenation.
2. Remove any lines that begin with an asterisk (comments).
3. Combine the remaining single-line records into one multiline record.
4. Display the output stream on the NetView screen.

The remainder of this unit discusses each of these PIPE stages and more.

Asynchronous command responses



Asynchronous command responses

- Some commands produce asynchronous output
 For example, MVS commands
- Requires CORRWAIT stage for correlating output
 - PIPE MVS D A,L | CONS results in no output
 - PIPE MVS D A,L | CORR 5 | CONS

```
" AOFDA
IEB1114I 10.11.18 2011.164 ACTIVITY 147
JOBS      M/S   TS USERS   SYSAS   INITS   ACTIVE/MAX VTAM   OAS
00003     00024   00001   00033   00013   00001/00040   00017
CANACN    CANACN  CNDL   NSW   S   LLA   LLA   NSW   S
JES2      JES2    IEFPROC  NSW   S   VLF   VLF   NSW   S
VTAM      VTAM   VTAM   NSW   S   DLF   DLF   NSW   S
RACF      RACF   RACF   NSW   S   TSO   TSO   OWT   S
SDSF      SDSF   SDSF   NSW   S   TCP/IP  TCP/IP  NSW   S0
TN3270    TN3270  TN3270  NSW   S0  DB9GMSTR DB9GMSTR IEFPROC  NSW   S
HTTPD1    HTTPD1  WEBSRV1 IN    SO  NFSS   NFSS   GFSAMAIN NSW   S0
IBMSM    IBMSM   ISM13  NSW   SO  DE9GIRLM DB9GIRLM  NSW   S
DB9GDBM1 DB9GDBM1 IEFPROC  NSW   S  DB9GDISP DB9GDISP IEFPROC  NSW   S0
PORTMAP  PORTMAP EMAP   OWT   SO  OSNMPD OSNMPD  OWT   S0
SNMPQE  SNMPQE  SNMPQE OWT   SO  FTPD1  STEP1   FTPD   OWT   AO
INETD4   INETD4  OMVSKERN OWT   AO  SSHD4  STEP1   START2  OWT   AO
NETVSSI  NETVSSI NETVIEW NSW   S   NETVIEW NETVIEW NSW   S0
CNMEUNIX CNMEUNIX *OMVSEX OWT   SO ...
```

1-15
IBM Software Group | Tivoli Software
 © 2011 IBM Corp.

Some command responses are *asynchronous* responses. The output of the command does not return to the pipeline without the use of PIPE stages to wait for the response. An example would be a two-stage PIPE that consists of the MVS stage to issue a command and the CONSOLE stage to display the pipeline data. No data would result because of the asynchronous response from the MVS command.

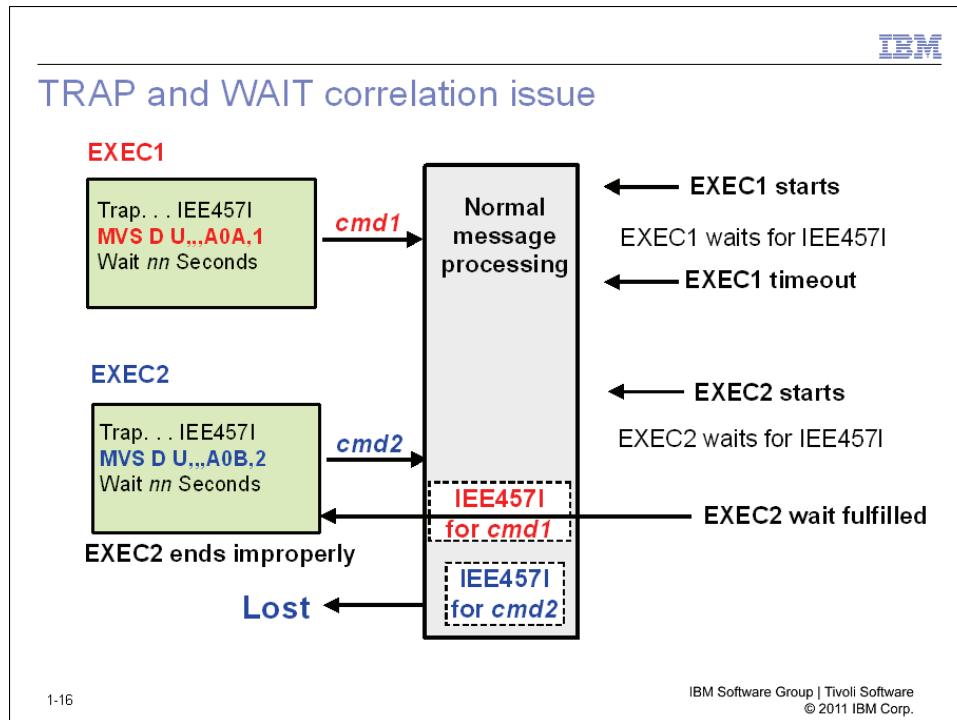
To collect and correlate asynchronous data, the CORRWAIT (WAIT for short) stage must be specified also. It can specify a maximum time to wait for the data



Note: If you do not specify a minimum CORRWAIT time, the PIPE waits one second. If you code CORR *, the PIPE waits forever.

In most cases, you wait for the asynchronous data and specify a *terminating condition* to end the wait as soon as the condition is met. When the PIPE is in a wait state, the task that runs the PIPE also waits. No commands that are queued to the task run until the PIPE completes.

TRAP and WAIT correlation issue



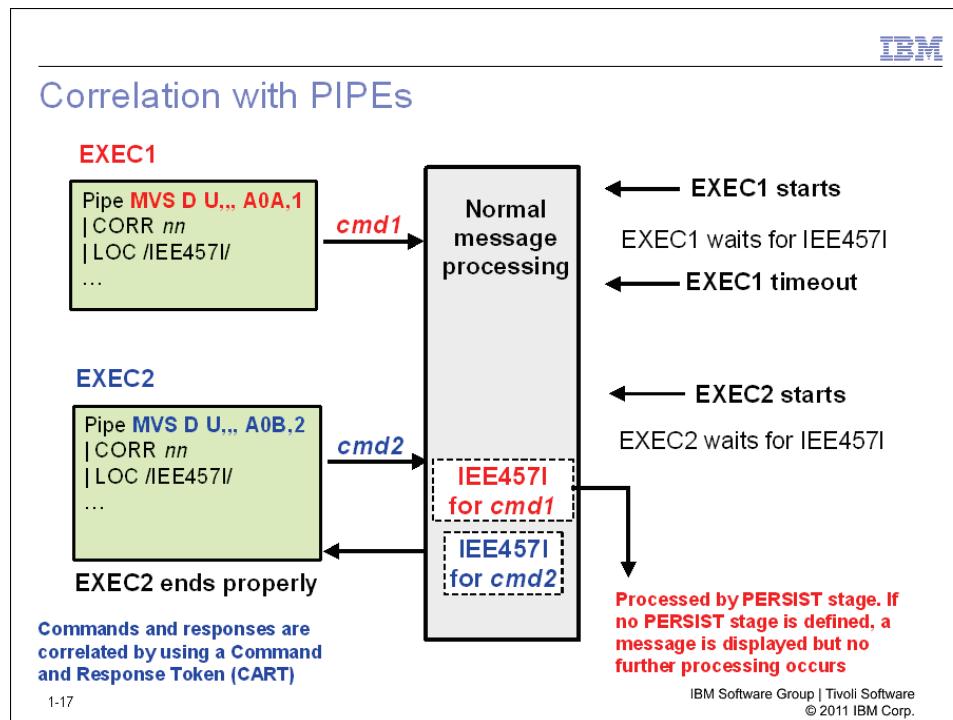
Suppose you have two EXECs that issue the MVS **D U** command. Each EXEC traps the IEE457I message in response to the command. This slide shows a scenario as follows:

1. The first EXEC, EXEC1, encounters a WAIT timeout and ends.
2. EXEC2 is issued and waits for message IEE457I.
3. When the first EE457I message arrives, it is intended for EXEC1.
4. EXEC2 is active and encounters an error because of the erroneous IEE457I message.
5. After EXEC2 ends, the IEE457I that is intended for it arrives and is then discarded.

This scenario is common when writing REXX EXECs to TRAP and WAIT for the same message ID on the same task. Two solutions are as follows:

- You can add code to the EXECs to test the message to make sure it is the correct one.
- You can use a PIPE command similar to the one shown on the next slide.

Correlation with PIPEs



With command correlation, every command that follows the pipeline has a Command and Response Token (CART) appended to it. Only messages that match the CART route to the output stream for a command. This enables the relevant response to be identified and returned asynchronously. Only the correctly correlated replies enter the appropriate pipe.

In the example, the LOCATE stage confirms that each PIPE is waiting for the IEE457I message that pertains to the MVS command issued. The PERSIST stage can process messages that arrive after the PIPE ends. By default, the PERSIST stage displays the message on the task where the PIPE ran. Discussion about PERSIST occurs later in this unit.

Using NetView pipelines resolves two issues as follows:

- All IEE457I messages are correlated and return to the correct pipeline.
- The IEE457I that arrives after EXEC1 completes processes based on the PERSIST stage specification.

When correlation is necessary



When correlation is necessary

- Synchronous responses: Not necessary
Local NetView commands
- Asynchronous responses: Necessary
 - MVS commands
 - VTAM commands
 - UNIX commands
 - TSO commands
 - Commands to remote NetView
- Using CORRWAIT or CORRCMD to wait for responses
- Specifying terminating condition (TOSTRING or TAKE) to end wait
- If no terminating condition, PIPE waits until timeout

1-18

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You should add correlation (CORRWAIT) to your pipelines when issuing the following types of commands:

- Commands to a remote NetView
- Most MVS commands
- VTAM commands
- TSO commands
- UNIX® commands



Note: If you do not specify a minimum CORRWAIT time, the PIPE waits one second. If you code CORR *, the PIPE waits forever.

When the PIPE is in a wait state, the task that runs the PIPE also waits. No commands that are queued to the task run until the PIPE completes.

Correlation notes



Correlation notes

- Not all NetView commands can be correlated
- Some subsystem and MVS commands do not return a correlated response
- If a command is timer-scheduled, output is not trapped in the PIPE that issues the timer command

1-19

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Not all commands are capable of correlation. A subset of older NetView commands are not correlatable. You can test if a command is correlated with a simple PIPE as follows:

```
PIPE NETV command_name | HOLE
```

If you see output for *command_name*, it is not correlated. Correlated command output passes to the HOLE stage and is suppressed. Discussion about the HOLE stage occurs later in this unit.

Lesson 2: PIPE stages



Lesson 2: PIPE stages

- PIPE syntax
- Interfacing with applications
- Interacting with users
- Storing data
- Filtering data
- And more

1-20

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This lesson discusses the PIPE command syntax and several of the most commonly used PIPE stages.

Online help as follows is available when you are logged on to NetView:

- **HELP PIPE SYNTAX:** Displays help for the syntax of the PIPE command.
- **HELP PIPE STAGES:** Displays a list of PIPE stages. Select a stage in the list to see the help.
- **HELP PIPE *stage_name*:** Displays the help for the specified stage. For example, HELP PIPE NETVIEW displays the help for the NETVIEW stage.

Simple PIPE command syntax

IBM

Simple PIPE command syntax

```

>>> PIPE ----->

  .- STAGESEP | -----.
  |-----+-----+-----+-----+
  '- STAGESEP value-'  '- ESC value-'  '- END value-' 

  . | ----- .
  v |
>>>-----+-----+-----+-----| Stages |-----><
      '- label: -' '-' (DEBUG) -
  
```

- PIPE can be issued from command line or a procedure
- Many stages supported:
 - Issue commands
 - Read files
 - Parse data records in pipeline
 - And more

1-21

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide shows the basic PIPE command syntax. PIPE stages are separated by a character called a *stage separator*. For example, **PIPE NETVIEW LIST DSMLOG | LOC ... | ... | CONSOLE** uses the default stage separator as follows:

- With the NETVIEW stage to issue a LIST DSMLOG command
- With the LOC stage to filter data
- With the CONSOLE stage to display the results

You can use the **STAGESEP** option to specify the character to use for separating the stages within a pipeline. The default stage separator is the character x'4F'. Depending on the code page used by your workstation, this stage separator is a solid vertical bar (|), split vertical bar (|), or exclamation mark (!).

The **ESC** option indicates that the character that follow the specified character is treated literally when the pipeline specification is parsed. Alternatively, you can use the stage separator character to self-escape itself. Two side-by-side separators resolve to one such character taken literally.

The **END** option identifies a character that is used for signifying the end of a pipeline.

Simple example of a complex pipeline

IBM

Simple example of a complex pipeline

NAMES	
BOB	07/22/2007 ...
TOM	06/01/2007 ...
PHIL	09/06/2007 ...
BOB	08/15/2007 ...
SUE	09/15/2007 ...
.	.
.	.

```
PIPE      (END %)      < NAMES
| A: LOCATE /BOB/
| COLOR RED
| CONSOLE
%A:
| COLOR BLUE
| CONSOLE
```

1. Read data NAMES file
DSIPARM member
2. LOCATE all lines containing string BOB
Display on console in red
3. All other lines: Display in blue

1-22

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide shows a simple example of a *complex pipeline*, a series of simple pipelines that are connected with labels. Some stages accept multiple input streams and others generate multiple output streams. In most cases, when you see an END character that is defined in a pipeline, it is a complex pipeline.

Starting with a simple pipeline, the NAMES file is read (from the NetView DSIPARM data set, by default) with the **PIPE** < stage. The **LOCATE** stage filters out all records that do contain the string BOB and displays them in red on the NetView operator screen.

Using the END character (%), you can specify the end of a pipeline. All output that the first pipeline filters passes to the second pipeline. The NetView operator screen displays the data that is in the second pipeline in blue.

Discussion about complex pipelines occurs later in this unit.

Interfacing with applications

IBM

Interfacing with applications

- **NETView:** Run a specified NetView command
- **MVS:** Run a specified MVS command
- **TSO:** Run a specified TSO command by using the NetView TSO command server
- **UNIX:** Run a specified UNIX command by using the NetView UNIX command server
- **PPI:** Pass data to a PPI receiver queue

Need **CORRWAIT** with MVS, TSO, and UNIX stages

1-23 IBM Software Group | Tivoli Software
 © 2011 IBM Corp.

Using NetView pipelines, you can issue NetView, MVS, VTAM, TSO, or UNIX commands as follows:

- **NETVIEW stage:** Specifies to run a NetView, MVS, or local VTAM command. An example is PIPE NETV LIST DSMLOG.
- **MVS stage:** Specifies to run an MVS command. The MVS stage can be used instead of coding MVS commands within the NETVIEW stage. An example is PIPE MVS D A,L.
- **TSO stage:** Routes a command to a NetView TSO server, which is a batch job submitted by NetView or a started task.
- **UNIX stage:** Routes a command to a NetView UNIX server where the command is run and the results returned.

You can also use the **PPI** stage to send and receive data from non-NetView applications by using the NetView program-to-program interface (PPI).

Interacting with users



Interacting with users

- **CONsole:** Display contents of pipeline
- **LITeral:** Insert text string into pipeline
- **LOGto:** Log contents of pipeline to DSLOG, system log, or hardcopy log
- **HELDmsg:** Read held-message queue into pipeline

1-24

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide shows the pipeline stages that can be used for interacting with an operator. For example, you can display a message to the operator with the CONSOLE stage.



Note: If you specify CONSOLE ONLY, messages are to be displayed but not held, logged, or exposed to automation. For example, use the ONLY option when sending text to the operator.

Working with files



Working with files

- **QSAM:** Read from (and write to) files
- <: Read data from file into pipeline
- >: Write data from pipeline to file

1-25

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide shows the pipeline stages that can be used for reading from and writing to members of partitioned data sets.

The members can also be loaded in NetView storage for quicker access by programs with the INSTORE stage. The members are then read from storage instead of from the disk when you use the QSAM or < stage. The in-storage members are also viewable when operators use the BROWSE command to view a member.

Storing pipeline data

IBM

Storing pipeline data

- **SAFE:** Read data from (and write to) local message queue
 - Default safe name: * (asterisk)
 - Data stored in *named* safes for use later within command group
- **KEEP:** Read data from (and write to) task-wide storage
 - Data stored in *named* keeps for use by other procedures
- **STEM:** Read from (and write to) stem variables
- **VAR:** Read from (and write to) variables
 - Local, task global, and common global

1-26

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This slide shows the pipeline stages that can be used for storing data. Data can be stored internally in *safes* or *keeps* or in variables and arrays.

A common pipeline practice is using REXX EXECs to issue commands (for example, NetView LIST DSLOG). The EXEC filters the pipeline contents, and stores the output stream in a stem variable. The EXEC can loop through the contents of the stem variable and parse to determine the log that was active, for example.



Note: You can use PIPE EDIT to perform the same logic without calling a REXX procedure.



Note: You cannot issue these stages from the NetView command line.

Filter stages (1 of 2)



Filter stages (1 of 2)

- Overview
 - Issue command, generating multiline response
 - Separate response into multiple single-line records
 - Filter desired text from response
 - Collect multiple single-line messages into multiline message
 - Pass data to output stream
- Data records can be single-line or multiline
 - **SEParate:** Break multiline message into single-line records
 - **COLLect:** Combine single-line records into multiline message
 Useful when logging or displaying messages

1-27

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Using pipeline filters for working with data, you can accomplish the following tasks:

- Separate multiline messages into multiple single-line messages (SEPARATE)
- Collect multiple single-line messages into a multiline message (COLLECT)

Filter stages (2 of 2)



Filter stages (2 of 2)

- Can select data records based on content
 - **LOCate**: Select records that match specified criteria
 - **NLOCate**: Discard records that match specified criteria
 - **TOString**: End data stream when specified character string found
- Can select record by position
 - **TAKE**: Specify number of records to remain in pipeline
 - **DROP**: Specify number of messages to discard from pipeline
- Can discard pipeline data
HOLE: Discard entire contents of pipeline
(Can be used to test if command can be correlated.)

1-28

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Several stages select pipeline messages to include or discard. They read all messages in the pipeline, but write only those that meet selection criteria to the stage output stream. Using selection filters, you can accomplish the following tasks:

- Keep all messages that contain a match for a specified text string (LOCATE).
- Discard all messages that contain a match for a specified text string (NLOCATE).
- Keep messages up to and including the message that contains a match for a specified text string (TOSTRING).
- Keep the first or last *nn* messages in the pipeline (TAKE).
- Discard the first or last *nn* messages from the pipeline (DROP).
- Empty the pipeline of messages after writing them to a variable or testing a command to see if it is correlated (HOLE).

TOSTRING, TAKE, and DROP stages generate a terminating condition for the CORRWAIT stage.

Issuing commands: NETView stage

IBM

Issuing commands: NetView stage

```
|-- NETVIEW --+-----+----->
' - (+-----+ +-----+ +-----+ ) - '
'--+CGI---+-' '-NOPANEL-' '-MOE-' 
'-ECHO-' 

>+-----+
' - cmd_text - '
```

Run NetView, local VTAM, or MVS command and place results in pipeline:

- **MOE** (*Message on Error*): Insert DWO369I and return code into pipeline after any resulting messages
- **ECHO**: Write command text to pipeline before command runs
- **CGI**: Specify for HTML output
- **NOPANEL**: Disallow display of full-screen panel. Error message is inserted in pipeline
- **Cmd_text**: Command and parameters

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

The **NETVIEW** stage (abbreviated as **NETV**) runs a NetView, MVS, or local VTAM command. Use the **NETV** stage to issue VTAM DISPLAY, VARY, and MODIFY commands in a local or remote domain. The resulting messages are placed in the pipeline.

Message on error (MOE) examines the return code from the command. If the return code is not zero, MOE inserts message DWO369I (containing the return code) into the stream after any messages that the command generates. If you do not specify MOE, return codes from commands are ignored.

When NOPANEL is specified, a full-screen panel display is disallowed. If a full-screen panel is encountered, message BNH113W is inserted into the pipeline and the command receives an error code.

You can place the **NETVIEW** stage anywhere in the pipeline specification. If **NETVIEW** is not the first stage, the command text is optional. The command runs once for each message that the previous stage delivers. Every time the command runs, the input message becomes the current message during the processing.

The **NETVIEW** stage does not require an output stream, meaning that **NETVIEW** can be a last stage. Also, if a stage that follows **NETVIEW** disconnects, the **NETVIEW** stage continues to process if it has an input stream.

Asynchronous responses are a subset of NetView command responses. To collect and correlate asynchronous data, the CORRWAIT (WAIT for short) stage must follow.

Issuing commands: MVS stage

IBM

Issuing commands: MVS stage

```
|--- MVS ---+-----+-----+-----+  
|   MOE- '   '- cmd_text -'  
|-----|-----|-----|-----|
```

- Run **MVS** command and place results in pipeline:
 - **MOE**: If the return code is not zero, it inserts message DWO369I containing the return code into the stream after messages that the command might have returned
 - **Cmd_text**: Command and parameters
- Only MVS commands that the extended multiple console support (EMCS) consoles issue are to be correlated
- MVS command responses are asynchronous
Use CORRWAIT stage to collect and correlate responses

1-30

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **MVS** stage specifies an MVS command to run. You can use the MVS stage instead of coding MVS commands within the NETVIEW stage. All rules that apply to an MVS command that is issued by using the NETVIEW stage also apply here. Any restrictions that apply to running an MVS command under PIPE NETVIEW also apply to PIPE MVS.

MVS system commands can be correlated only when issued from extended multiple console support (EMCS) consoles.

MVS command responses are *asynchronous*. To collect and correlate asynchronous data, the CORRWAIT (WAIT for short) stage must follow.

Simple PIPE example 1

IBM

Simple PIPE example 1

Pipe NETV QRYGLOBL COMMON VARS=CNMSTYLE.AUTO.* | CONS ONLY

BNH031I NETVIEW GLOBAL VARIABLE INFORMATION		
BNH103I	COMMAND ISSUED AT: 06/13/11 13:13:07	
BNH061I		
BNH032I	COMMON GLOBAL VARIABLES	
BNH033I	GLOBAL VARIABLE NAME: _____	
BNH043I		
BNH039I	CNMSTYLE AUTO. NVSCMOPREVISION	GLOBAL VARIABLE VALUE:
BNH039I	CNMSTYLE AUTO. SAVEDBMAINT	BNH0F01
BNH039I	CNMSTYLE AUTO. EMRAUTOM	BNH0F01
BNH039I	CNMSTYLE AUTO. AUTOOLP	BNH0F00N
BNH039I	CNMSTYLE AUTO. PATS.TCPPIP	AUTOPKTS
BNH039I	CNMSTYLE AUTO. XMONDBMAINT	BNH0F02
BNH039I	CNMSTYLE AUTO. TCP5M.TCPPIP	BNH0F0CPS
BNH039I	CNMSTYLE AUTO. NAPOLTS2	AUTOC2
BNH039I	CNMSTYLE AUTO. NAPOLTS3	AUTOC3
BNH039I	CNMSTYLE AUTO. NAPOLTS4	AUTOC4
BNH039I	CNMSTYLE AUTO. NAPOLTS4A	BNH0F04
BNH039I	CNMSTYLE AUTO. SMONDBMAINT	
BNH039I	CNMSTYLE AUTO. OPKT.TCPPIP	BNH0F0PKT
BNH039I	CNMSTYLE AUTO. COLTSK5	BNH0F0CT5
BNH039I	CNMSTYLE AUTO. XCEHISSC	BNH0F0XSC
BNH039I	CNMSTYLE AUTO. TCPDBMAINT	BNH0F02
BNH039I	CNMSTYLE AUTO. MEMSTORE	AUTO2
BNH039I	CNMSTYLE AUTO. ENDURTA	BNH0F0DRT
BNH039I	CNMSTYLE AUTO. NVSORPTSK	AUTONVSP
BNH039I	CNMSTYLE AUTO. IDLEOFF	AUTOF1
BNH039I	CNMSTYLE AUTO. DLMRPT	BNH0F0CE
BNH039I	CNMSTYLE AUTO. COLTSK7	BNH0F0CT7
BNH039I	CNMSTYLE AUTO. AP-SERV	BNH0F0MSI
BNH039I	CNMSTYLE AUTO. MASTER	AUTOF1
BNH039I	CNMSTYLE AUTO. NALCLOP	BNH0F0ALC
BNH039I	CNMSTYLE AUTO. POLICY	BNH0F0RN
BNH039I	CNMSTYLE AUTO. TCPCONN.TCPPIP	BNH0F0CPC
BNH039I	CNMSTYLE AUTO. NETCONN	AUTOF2
BNH039I	CNMSTYLE AUTO. DLARHTO	AUTOF2
BNH039I	CNMSTYLE AUTO. COLTSK6	AUTOC6
BNH039I	CNMSTYLE AUTO. PRIMARY	AUTOF1
BNH039I	CNMSTYLE AUTO. FKOPKTS	BNH0F0SRV
BNH035I	NUMBER OF VARIABLES FOUND: 32	

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The first example is a simple pipeline that issues the **QRYGLOBL** command to query all NetView common global variables that begin with the **CNMSTYLE.AUTO.** string. The output is to be displayed to only the console. Data in the pipeline can be filtered with LOC, NLOC, TAKE, DROP, and so on.

Simple PIPE example 2

IBM

Simple PIPE example 2

Pipe MVS D A,L | CORRWAIT 10 | CONS

```
NetView V6R1MD      Tivoli NetView     AOFDA PHOLM    06/13/11 13:20:36 U
* AOFDA    PIPE MVS D A,L | CORRWAIT 10 | CONS
" AOFDA
IEE114I 13,20,36 2011,164 ACTIVITY 461
JOBS   M/S   TS  USERS    SYSAS  INITS   ACTIVE/MAX VTAM   O&S
00003  00024  00001  0003  00013  00001/00040  00017
CANACN CANACN  CNDL  NSW   S    LLA    LLA    NSW   S
JES2   JES2   IEFFPROC  NSW   S    VLF    VLF    NSW   S
VTAM   VTAM   VTAM  NSW   S    DLF    DLF    NSW   S
RACF   RACF   RACF  NSW   S    TSO    TSO    STEP1  OUT  S
SDSF   SDSF   SDSF  NSW   S    TCP/IP TCP/IP  NSW   SO
TN3270 TN3270  TN3270 NSW   SO DB9GMSTR DB9GMSTR IEFFPROC NSW   S
HTTPD1 HTTPD1  WEBSRV1 IN    SO NFSS   NFSS   GFSAMAIN NSW   SO
IBMSM  IBMSM  ISM13  NSW   SO DB9GIRL1 DB9GIRL1 NSW   S
DB9GDBM1 DB9GDBM1 IEFFPROC  NSW   S    DB9GDIST DB9GDIST IEFFPROC NSW   SO
PORTMAP PORTMAP  PMAP  OUT   SO CSNMPD  CSNMPD  CSNMPD  OUT  SO
SNMPQE SNMPQE  SNMPQE OUT   SO FTPD1  STEP1   FTPD   OUT  AO
INETD4 INETD4  STEP1  OMVSKERN OUT   AO SSHD4  STEP1   START2  OUT  AO
NETVSSI NETVSSI  NETVIEW NSW   S    NETVIEW NETVIEW NETVIEW NSW   SO
CNMEUNIX CNMEUNIX *OMVSEX OUT   SO
*LOGON* OUT
```

With no terminating condition, the PIPE waits for the full 10 seconds

Add a terminating condition:

Pipe MVS D A,L | CORRWAIT 10 | **TOSTRING /IEE114I/** | CONS

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This example issues the **D A,L** command by using the **MVS** stage. MVS command responses are *asynchronous* and require the addition of the **CORRWAIT** stage to wait (in this case, 10 seconds) for all responses. The **IEE114I** multiline message is to be displayed as soon as it is received, but you wait for 10 seconds for any additional messages.

To end the wait when the **IEE114I** arrives, use a PIPE stage to generate a *terminating condition*. For example, use **TAKE 1** or **TOSTRING /IEE114I/**. If you do not use the **CORRWAIT** stage, you do not see a response.

Simple PIPE example 3

IBM

Simple PIPE example 3

PIPE (STAGESEP .) MVS D A,L . CORRWAIT 10

- . LITERAL /Command to MVS/
- . COLLECT
- . CONS

```

NetView V6R1MO      Tivoli NetView   AOFDA PHOLM   06/13/11 13:40:35
* AOFDA    PIPE (STAGESEP .) MVS D A,L . CORRWAIT 10 . LITERAL /COMMAND TO
      MVS/. COLLECT . COLLECT . CONS
| AOFDA
COMMAND TO MVS
IEE114I 13,40,25 2011,164 ACTIVITY 499
JOBS   M/S   TS  USERS   SYSAS   INITS   ACTIVE/MAX VTAM   OAS
00003  00024  00000  0003   00013  00000/00040  00017
CANACN CANACN  CNDL   NSW S  LL A   LL A   LLA   NSW S
JES2   JES2   IEFFPROC NSW S  VLF   VL F   VL F   NSW S
VTAM   VTAM   VTAM   NSW S  DL F   DL F   DL F   NSW S
RACF   RACF   RACF   NSW S  TSO   TSO   STEP1  OUT  S
SDSF   SDSF   SDSF   NSW S  TCP/IP TCP/IP TCP/IP NSW SO
TM3270 TM3270 TM3270 NSW SO DB9GMSTR DB9GMSTR IEFFPROC NSW S
HTTPD1 HTTPD1 WEBSEV1 IN  SO NFSS NFSS GFSAMAIN NSW SO
IBMSM  IBMSM  IBM13  NSW SO DB9GIRLM DB9GIRLM NSW S
DB9GDBM1 DB9GDBM1 IEFFPROC NSW S  DB9GDIST DB9GDBIST IEFFPROC NSW SO
PORTMAP PORTMAP PI MAP OUT  SO CSNMPD CSNMPD OSNMPD OUT SO
SNMPQE SNMPQE SNMPQE OUT  SO FTTP1 STEP1 FTPI  OUT AO
INE1D4 STEP1 OHVSKERN OUT AO SSHD4 STEP1 START2 OUT AO
NETVSSI NETVSSI NETVIEW NSW S  NETVIEW NETVIEW NETVIEW NSW SO
CNMEUNIX CNMEUNIX *OMVSEX OUT  SO

```

→

1-33

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This example issues the **D A,L** command by using the **MVS** stage with a *stage separator* of a period (.). The **CORRWAIT** stage is used for waiting for the asynchronous response.

The **LITERAL** stage inserts a text string into the pipeline when the CORRWAIT stage ends. Collect text string and IEE114I into a multiline message by using the **COLLECT** stage. There is no terminating condition. The output is not displayed until the CORRWAIT ends.

The text string is displayed first. Data that appends in the pipeline output displays as *last in, first out* (LIFO), by default. You can change the order by using the **REVERSE** stage.

MOE operand and error messages

MOE operand and error messages

If you specify **MOE** operand on a PIPE stage, test for message DWO369I

```
/* REXX */
'Pipe MVS (MOE) D A,L',
    '| CORRWAIT 10 | TAKE FIRST 1 | STEM MSGTEXT. '
Parse var msgtext.1 msgid msgstr
Select
    When msgid = 'DWO369I' then          /* Bad RC */
        /* Include code to parse DWO369I message      */
    When msgid = 'IEE114I' then          /* All OK */
        /* Code here to process the IEE114I MLWTO      */
    Otherwise
end
```



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The text of a DWO369I resembles the following text:

```
DWO369I  stagename STAGE (stagenum) HAD RETURN CODE retcode
```

Your procedure can display the DWO369I to users to inform them of the error.

CORRWAIT stage

IBM

CORRWAIT stage

The MVS **DISPLAY TCPIP** command issues two messages: EZAOP41I and EZAOP50I

Example PIPE to suppress the EZAOP41I:

```
/* REXX Example */
'Pipe MVS D TCPIP',          /* issue D TCPIP command */
  '| CORR 10',                /* wait max 10 seconds */
  '| TAKE FIRST 1',           /* take first message */
  '| CONSOLE'                 /* display on console */

" AOFDA
EZAOP50I TCPIP STATUS REPORT 510
COUNT      TCPIP NAME    VERSION      STATUS
-----      -----
1          TCPIP        CS V1R12     ACTIVE
*** END TCPIP STATUS REPORT ***
```

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **CORRWAIT** stage allows asynchronous messages (generated by the previous stage) to be returned to the pipeline, and sets a specified time limit if needed. Asynchronous messages are those that return to the NetView program from commands running in another application.: Example commands are MVS, VTAM, TSO, UNIX, and commands running at another NetView. When asynchronous responses are to result from a command issued in a pipe, the next stage command must be CORRWAIT. Without CORRWAIT, the messages are lost.

If a message arrives after the time interval expires, it is discarded. In some instances, a very large time interval might be needed. The wait time can be long between a message arrival and time interval expiration.

A *terminating condition* to end the wait state should follow a CORRWAIT stage. **TAKE** and **TOSTRING** are example stages that end the wait. Each causes the **CORRWAIT** stage to disconnect. CORRWAIT without a *terminating condition* (for example TAKE) waits until the timeout value for all messages.



Note: LOCATE does not cause a terminating condition.

In example on the slide, the **MVS DISPLAY TCPIP** command is issued with a CORRWAIT of 10 seconds. The **TAKE FIRST 1** defines a terminating condition that takes control when the first message arrives (if fewer than 10 seconds). Discussion of the TAKE stage occurs later in this unit.

In this example, the second message, EZAOP41I, message is suppressed:

```
EZAOP41I 'DISPLAY TCPIP' COMMAND COMPLETED SUCCESSFULLY
```

CORRWAIT can end under the following conditions:

- Operator issues GO command.
- PIPE issues GO command.
- Match is found for expected condition in a filter stage, for example, TOSTRING or TAKE stages.
- CORRWAIT stage timeout occurs.
- PIPEND stage occurs.
- Secondary input stream disconnects.
- Message is received.

The wait indicator (**W**) is displayed on the operator screen when CORRWAIT waits for messages.



Note: When the PIPE is in a wait state, the task that runs the PIPE also waits. Commands queued to the task do not run until the PIPE completes.

CORRCMD stage

IBM

CORRCMD stage

```
VARY DSICCDEF
  * VTAM vary command
  CORRWAIT 18
  * PERSIST 120 MINUTES ROUTE AUTHRCVR
  TOSTRING LAST 1.7 /IST314I/ 1.7 /IST093I/ 1.7 /IST061I/ 1.8 /IST1132I/,
    1.7 /IST039I/ 1.7 /IST453I/ 1.7 /DWO369I/ 1.7 /IST073I/ 1.8 /IST1149I/,
    1.7 /IST607I/ 1.8 /IST1133I/ 1.7 /IST455I/ 1.7 /IST072I/,
    1.8 /IST1264I/ 1.8 /IST1149I/ 1.7 /IST455I/ 1.7 /IST607I/ 1.7 /IST105I/
```

```
/* REXX */
/* PIPE to issue a VTAM VARY NET,ACT command using DSICCDEF */
/* instead of coding extensive list of terminating conditions. */
'Pipe CORRCMD NETV V NET,ACT,ID=xyz', /* issue VARY command */
  '| CORRWAIT 18',
  '| TOSTRING LAST 1.7 /IST314I/ ... ',
  '| CONSOLE' /* display on console */
```

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

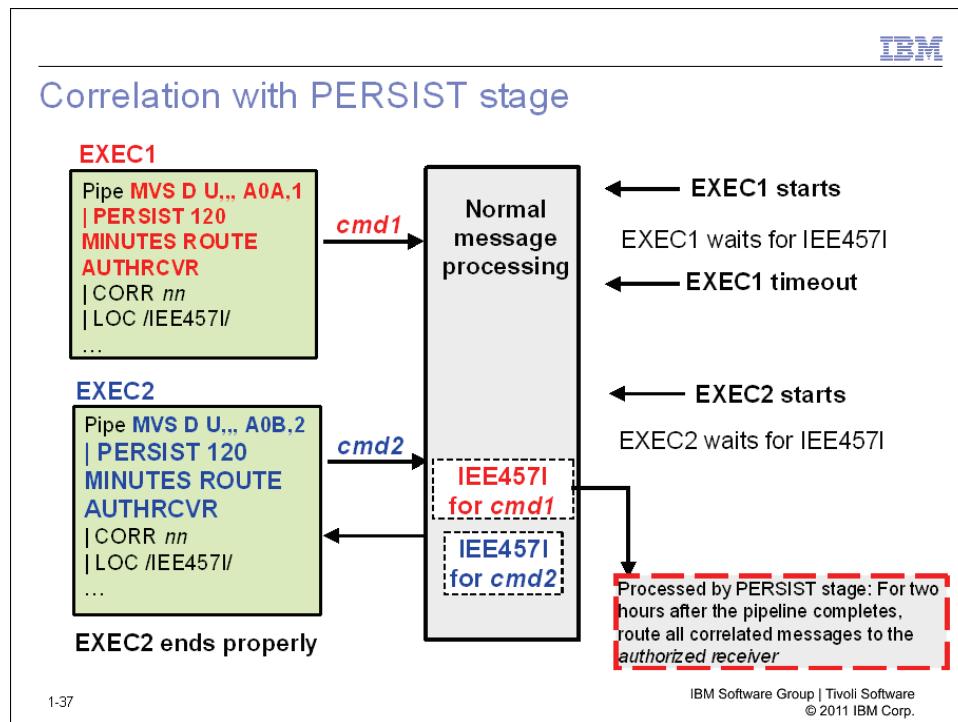
The **CORRCMD** stage (abbreviated as **CC**) processes a command and inserts appropriate correlation wait and termination condition stages. It gathers data from the command being processed. CORRWAIT is included in the inserted stages.

The timeout interval and termination conditions to use are defined in NetView DSIPARM member **DSICCDEF**. A sample file named DSICCDEF (CNMS1082) is provided with NetView. This member contains several commands with appropriate timeout and termination stages. Issue HELP CCDEF or CCDEF QUERY=*. NetView commands to display information pertaining to these settings.

This slide shows how to use the sample definitions defined in DSICCDEF for the VTAM VARY command. DSICCDEF defines the CORRWAIT and TOSTRING stages. When a PIPE CORRCMD stage is encountered for a VTAM VARY command, the CORRWAIT and TOSTRING stages append into the pipeline.

To include the PERSIST stage, uncomment the **PERSIST 120 MINUTES ROUTE AUTHRCVR** statement. To simplify the coding of your pipeline stages, use DSICCDEF and CORRCMD

Correlation with PERSIST stage



The **PERSIST** stage defines the action to perform when messages arrive after termination of the correlation that a CORRWAIT stage represents. The conditions that PERSIST defines are enabled after termination of the preceding CORRWAIT stage (WAIT) when one of these conditions is true:

- WAIT times out.
- WAIT responds to GO or RESET.
- WAIT ends prematurely because of a PIPEND (non-zero return code).
- DEBUG option occurs for the PERSIST stage. The PERSIST stage activates when the DEBUG option is specified.

If you specify more than one PERSIST stage for a command correlation environment, only the last specified PERSIST stage takes effect.

Messages subject to the DISPLAY action are exposed to user exits, trapping, automation, and logging. For more information about exposure, see “PIPE EXPOSE stage” on page 1-101.

Use the **LIST PERSIST** command to display the status of all enabled PIPE PERSIST elements or use the **STOP PERSIST** command to end an enabled persist.

Messages subject to ROUTE action are routed first, then are exposed as for other messages.

A message subject to COMMAND action is provided as the current message when the indicated command runs. Any output from the command, including the original message, is subject to exposure in the same way as the output of a command issued from the command facility command line. When PERSIST invokes a command, it does so with the same authority as was in effect for the pipeline which established the PERSIST action.

Student exercise

Student exercise



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

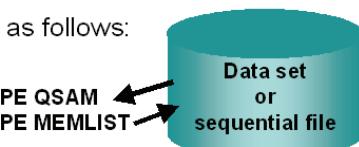
Open your *Student Exercises* book and perform Exercises 1 and 2.

Lesson 3: Using PIPEs to access files

IBM

Lesson 3: Using PIPEs to access files

- **QSAM** stage reads from and writes to areas as follows:
 - Partitioned data sets (PDS)
 - Sequential files
- File name can be one of two forms:
 - Fully qualified data set name, including member name
 - Data definition (DD) name from JCL or ALLOCATE command
- Files do not need to be allocated to NetView
- From-disk (<) and to-disk (>) stages are specialized for NetView files
- You use the **MEMLIST** stage to display list of members in NetView DD or data set



1-39

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **QSAM** stage reads and writes from dynamically allocated data definition (DD) names or data sets. Other devices are also supported when allocated for physical sequential access. The < and > (synonyms for QSAM read and write) can be used only when < and > are immediately followed by a data set name enclosed in quotation marks.

When QSAM is the first stage in a pipeline, the file is read into the pipeline. Otherwise, the contents of the pipeline are written to the file.

The < (from-disk) stage must be the first stage of pipeline. The from-disk stage is limited to a subset of data sets that are allocated to NetView. Issue **BROWSE !** to display the list.



Note: For NLS users, you can use X'5A' to represent the exclamation point character (!).

The **MEMLIST** stage creates a list of members in one or more partitioned data sets (PDS) or data definitions (DD). For a DD, the members are listed for each data set in the concatenation. Members that are defined in an operator data set and members that are defined by using INSTORE COMMON are also listed.

PIPE QSAM stage

The screenshot shows a slide titled "PIPE QSAM stage". At the top right is the IBM logo. Below the title is a red rectangular box containing a code snippet:

```
I-- QSAM -----+-----+-----+-----+  
     +- (DD) ---+ '- data_definition_name-'  
     '- (DSN) -'
```

Below the code snippet is a description: "General purpose read and write utility". A bulleted list follows:

- QSAM stage reads from and writes to files:
 - Read when first stage
 - Write otherwise
- Output is series of single-line messages
- Data set name does not need quotes

At the bottom left is the page number "1-40". At the bottom right is the copyright notice: "IBM Software Group | Tivoli Software
© 2011 IBM Corp."

You can use the **QSAM** stage with either a data definition (DD) name that is defined by the ALLOCATE command, or a fully qualified data set name. You can enclose a data set name in single quotes. The quotes are ignored.

When specified as a first stage, QSAM reads from the data definition name or data set. When not specified as a first stage, QSAM writes to the data definition name or data set. The messages received on the input stream pass to the output stream.

PIPE < and PIPE > stages

These stages are specialized versions of PIPE QSAM for NetView files. The default data set name is DSIPARM. Additional parameters are supported. INCL reads all included members and resolves any *Data REXX* statements. DISKONLY reads members from disk only. In some cases, files are readable into NetView storage. DISKONLY does not read the in-storage copy.

When you use a *data_definition_name* of *, NetView searches all standard DD names, if allocated, for the specified member name in the following order:

1. DSICLD
2. DSIPARM
3. DSIPRF
4. DSIVTAM
5. DSIMSG

6. CNMPNL1
7. BNJPNL1
8. BNJPNL2
9. DSILIST
10. DSIOOPEN
11. DSIASRC
12. DSiarpt

Issue **BROWSE !** to display the list of supported standard DD names. For NLS users, you can use X'5A' to represent the exclamation point character (!).

Security considerations

Accomplish security for PIPE QSAM, <, and > stages by implementing security for **READSEC** and **WRITESEC** commands. For example, operator NETOP1 can read from DSIPARM but is not permitted to write to DSIPARM. Implement the stage as follows:

- Issue the READSEC DD=DSIPARM to test read access:

```
DSI633I READSEC COMMAND SUCCESSFULLY COMPLETED
```

- Issue the WRITESEC DD=DSIPARM to test write access:

```
BNH234E 'NETOP1' IS NOT AUTHORIZED TO USE KEYWORD 'DSIPARM'  
BNH235E THE KEYWORD 'DSIPARM' IS PROTECTED BY COMMAND IDENTIFIER  
'*.*.WRITESEC.*' IN 'TBLNAME=SECQSAM'  
DSI213I ACCESS TO 'DSIPARM' IS NOT AUTHORIZED
```

Example Command Authorization Table (CAT) statements for this example are as follows:

```
* Define oper group  
GROUP NVOPS1 NETOP1,NETOP2,OPER1,OPER2,OPER3  
* Restrict access to READSEC  
PROTECT *.*.READSEC.DSIPARM.*  
* Allow NVOPS1 to READSEC  
PERMIT NVOPS1 *.*.READSEC.DSIPARM.*  
  
* Restrict access to WRITESEC  
PROTECT *.*.WRITESEC.*  
PROTECT *.*.WRITESEC.*.*  
* No PERMIT for WRITESEC = no writes allowed
```

PIPE QSAM example

IBM

PIPE QSAM example

Read CNMCMGU (user command definitions)

- PIPE QSAM USER.DSIPARM(CNMCMGU) | COLL | CONS

```
| AOFDA
*** File updated at 08:38:03 on 14 Jun 2006 ***
DDEF.EZLE600A.CMDSYN =  TIMER,TIMERS,TIMR
%INCLUDE MYCMD
```

- Alternatives as follows:
 - PIPE <'USER.DSIPARM(CNMCMGU)' | COLL | CONS
 - PIPE < DSIPARM.CNMCMGU | COLL | CONS
 - PIPE < CNMCMGU | COLL | CONS

1-41

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

In the following example, the pipe reads member CNMCMGU explicitly from the USER.DSIPARM data set. An included member is MYCMD, but the QSAM PIPE displays only the member name.

```
PIPE QSAM USER.DSIPARM(CNMCMGU) | COLL | CONS
```

For comparison, this is a section of a simple REXX EXEC to read members.

```
'TRAP AND SUPPRESS ONLY MESSAGES'      /* TRAP/SUPPRESS MSGS */
'ALLOCATE DA('DataSetName') SHR FREE'   /* ALLOC/CONNECT FILE */
'WAIT FOR MESSAGES'                   /* WAIT FOR MESSAGES */
'MSGREAD'                           /* READ A MESSAGE IN */
'TRAP NO MESSAGES'                  /* DISABLE TRAP MSGS */
/* CNM272I ddname IS NOW [ALLOCATED | DEALLOCATED]
If MSGID() = 'CNM272I' then          /* allocate worked? */
  DO                                /* PROCESS CNM272I MSG */
    DDNAME = MSGVAR(1)               /* SAVE DYNAMIC DDNAME */
    ADDRESS MVS 'EXECIO * DISKR' DDNAME '(STEM Output.'
    Say 'File' DataSetName'/'ddname 'has' output.0 'lines:'
    Do i=1 to Output.0              /* Output loop */
      SAY SUBSTR(Output.i,1,68)       /* DISPLAY LINE TO USER */
    End                               /* Output loop */
    Say 'TYPE EXEC is now finished'
  END                                /* PROCESS CNM272I MSG */
ELSE                                /* MSG IS CNM272I */
  SAY MSGID() MSGSTR()              /* DISPLAY MESSAGE */
```

You can use one PIPE statement to replace this REXX program.

Example of PIPE <

IBM

Example of PIPE <

Read CNMCMDU and all included files:
PIPE < CNMCMDU INCL DISKONLY | COLL | CONS

```
| AOFDA
*** File updated at 08:38:03 on 14 Jun 2006 ***
DDEF.EZLE600A.CMDSYN = TIMER,TIMERS,TIMR
* Member MYCMD
CMDDEF.MINE.MOD=DSICCP
CMDDEF.MINE.ECHO=N
CMDDEF.MINE.TYPE=R
```

1-42

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

In the example, the pipe reads CNMCMDU, using the from-disk stage. The standard NetView DD names are searched and the member found in the DSIPARM library concatenation. The included member (MYCMD) is also read into the pipeline because the INCL option is specified on the from-disk stage.

PIPE < CNMCMDU INCL DISKONLY | COLL | CONS

MEMLIST stage example

IBM

MEMLIST stage example

- MEMLIST lists members in NetView DD name or data set
Output: Multiline message, one line per member
- Example: Display all NetView command lists and EXECs
PIPE MEMLIST DSICLD | CONS

```
CNME1505 0
CNME1096 0
FKXEDVPT 0
FKXEDVPA 0
...
SETCG    1
TSTCONN  1
...
CNME1048 6
CNME1049 6
```

0: File loaded in storage

1: First data set in DSICLD concatenation

6: Sixth data set in DSICLD concatenation

1-43

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **MEMLIST** stage lists members in NetView DD name or data set. The output is a multiline message with one line for each member as follows:

- Columns 1 to 8: Member name
- Column 10: Relative data set number
 - 1 if file is found in an operator data set
 - 0 if file is loaded in storage
 - 1 when a data set is specified

For DD name, number matches concatenation order shown by LISTA command

MEMLIST includes operator data sets and members that are loaded in storage by INSTORE and MEMSTORE.

Student exercise

IBM

Student exercise



1-44

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

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

Lesson 4: Pipelines and variables



Lesson 4: Pipelines and variables

- Pipeline output can be stored in variables
- When specified as the first stage, variable data can be read into the pipeline
- Stages for variables are as listed. Each can process local, task global, or common global variables
 - VAR
 - STEM
 - VARLOAD

1-45

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

With the **VAR** stage, you can read records from or write records to variables in a command procedure variable pool. The **\$VAR** stage is similar to VAR. However, \$VAR also reads or writes the VIEW attribute variables (which start with \$) that are associated with the specified data variables.

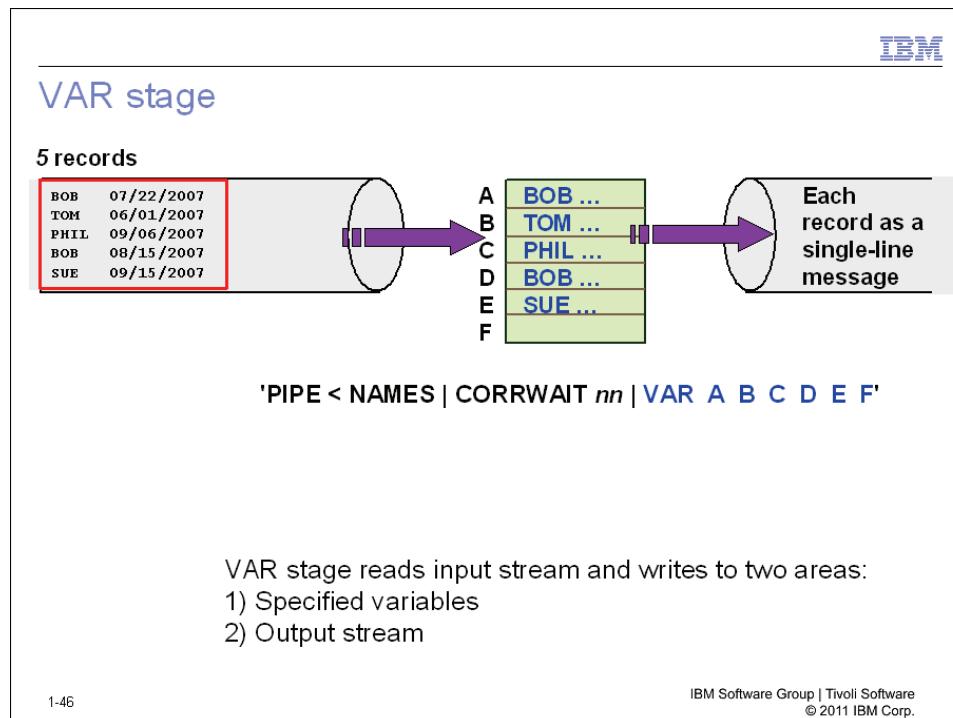
With the **STEM** stage, you can read or write an array of stemmed command procedure variables. The **\$STEM** stage is similar to STEM. However, \$STEM also reads or writes the VIEW attribute variables (which start with \$) that are associated with the specified array of stemmed data variables.

With the **VARLOAD** stage, you can set values for variables that pass in the input stream. The names and values of the variables that VARLOAD sets are specified by the records that passed on the primary input stream. VARLOAD sets one variable for each input message that contains a character other than a blank or asterisk in the first position of the record.



Note: You cannot issue these stages from the NetView command line.

VAR stage



The **VAR** stage reads data from the input stream, sets the specified variables, and writes the data to the output stream. In this example, the NAMES file is read into the pipeline. The VAR stage is passed five records, setting five variables A, B, C, D, and E. Variable F is not set because only five records were passed. The VAR stage syntax is as follows:

```
. - ----.
. - (0) -----. V      |
| --+--VAR---+-----+-----name-----+-----+
'-$VAR-'   +- (COMMON) -+
      +- (TASK) ---+
      '- (number) -'
```

When VAR is the first stage of a pipeline, records are read from the variable that is specified. Each record passes as a single-line message to the pipeline output stream.

When VAR is specified as a subsequent stage, messages are read from its input stream and written to both the specified variables and to its output stream. Data from the first input message is placed

in the first variable and written to the output stream. Data from the second message goes in the second variable and to the output stream. The data flows similarly for the rest of the process.

- (COMMON): Specifies to access the common global variable dictionary instead of the local dictionary for the procedure.
- (TASK): Specifies to access the task global variable dictionary instead of the local variable dictionary for the procedure.
- (*number*): Specifies the number of invocations (generations) to refer back when setting the variables. The default is zero (0).
- *name* - Specifies the name of the variable to read from or write to.



Tip: Pipeline variables are dropped before being used. This can cause errors if you use SIGNAL ON NOVALUE conditions in REXX programs. Use REXX SYMBOL() function to test for null variables: An example follows:

```
IF Symbol(E) <> 'LIT' then /* If variable E is not null */
```

PIPE VAR example

PIPE VAR example



Parse terminal name from **LIST "** command response:

First line: STATION: NETOP1 TERM: A01A701

```
/* REXX */

'PIPE NETV LIST '' ',      /* Issue LIST command */
' | TAKE FIRST   ',       /* take only line 1   */
' | VAR LINE1    ',       /* save in local var */

PARSE VAR LINE1 . 'TERM:' Oper_terminal

Say 'Your terminal is:' Oper_terminal
```

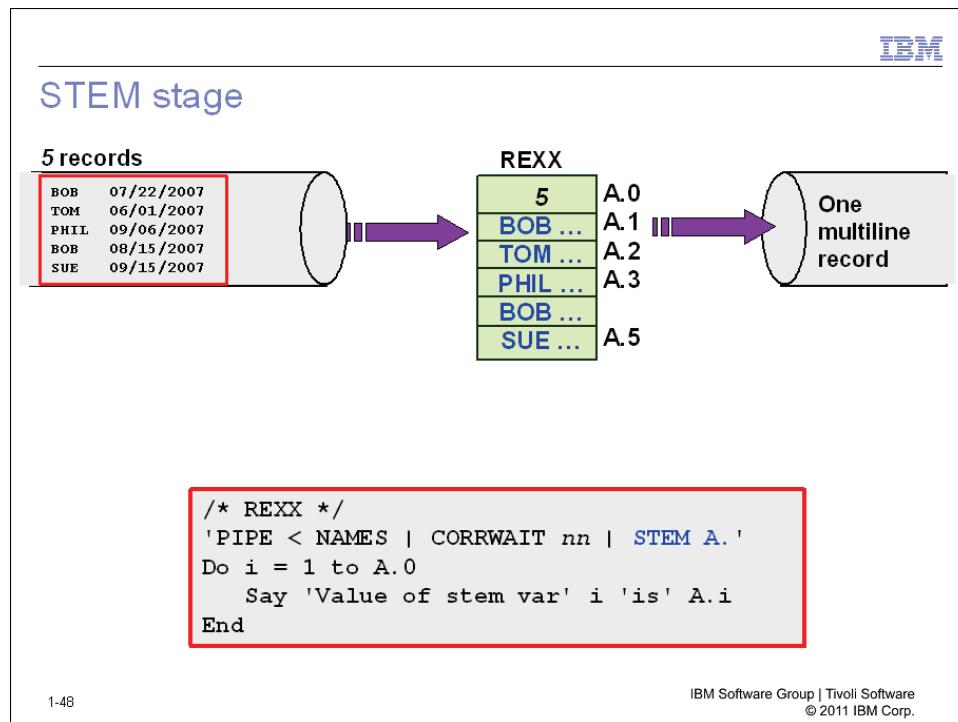
Example output: **Your terminal is: A01A701**

1-47

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

In this example, the NetView **LIST "** command is issued. Several single-line messages are generated. The pipeline takes the first message and places it into the LINE1 variable without stopping at the other messages. The next instruction, REXX **PARSE VAR**, parses LINE1 for the operator terminal.

STEM stage



The **STEM** stage reads data from the input stream, sets the specified stem variables, and writes the data to the output stream. In this example, the **STEM** output would be as follows:

```
Value of stem var 1 is BOB 07/22/2007
Value of stem var 2 is TOM 06/01/2007
Value of stem var 3 is PHIL 09/06/2007
Value of stem var 4 is BOB 08/15/2007
Value of stem var 5 is SUE 09/15/2007
```

- When STEM is the first stage, it reads records from an array of stemmed variables. Each record passes as a single-line message to the pipeline output stream.
- When STEM is used for reading records into a stem variable, the count of stem variables automatically updates in element zero.
- When STEM is not the first stage, it writes each line of each a message to stem variables and to the output stream. In addition, an integer appends to the given variable name (for example, VARNAME*n*).
- When STEM is used for writing records to the pipeline, the number of records to write must be defined in element zero.

The STEM stage syntax is as follows:

```
. - (0)-----.
>>+-----+-----+-----+-----+-----+-----+
    +-STEM--+- (number) --+-----+-----+-----+
    '-$STEM-'  +- (COMMON) --+
                  ' - (TASK) ---'

. - FROM 1-----.
>-+-----+-----+-----+-----+-----+-----+><
    +- FROM frnumber+-+-----+-----+-----+
    ' - APPEND-----'
```

- *stemroot*: Specifies the name of the stem variable to read from or write to. It should end with a period (.) if you use a REXX EXEC.
- APPEND: Specifies that new data should be appended as additional stem variables at the end of the current stem variables. APPEND can be used only on a stage that is not first.
- COLLECT: Causes STEM to build one multiline message instead of many single-line messages. COLLECT is allowed only when STEM is the first stage in a pipeline.
- FROM: Indicates a starting point for access to the stem variables.

Student exercise

Student exercise



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 4.

Lesson 5: Working with text in pipeline



Lesson 5: Working with text in pipeline

- COLOR
- LOCATE and NLOCATE
- TOSTRING
- TAKE and DROP
- COLLECT and SEPARATE
- CHOP
- BETWEEN
- REVERSE
- PICK
- SUBSYM
- CHANGE
- CASEI
- SPLIT
- STRIP
- SORT
- JOINCONT
- DUPLICAT
- DELDUPES
- COUNT

1-50

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This lesson contains information on several pipeline stages for modifying the contents of pipeline data. Use the NetView HELP PIPE STAGES command to display a panel with a list of all NetView pipe stages. On the panel, **tab over** to a stage name (for example, LOCATE) and press Enter to view the help for the LOCATE stage.

Manipulated pipeline data routes to the primary output stream of these pipe stages with all other data sent to the secondary output stream.

COLOR Stage: Change message attributes

IBM

COLOR stage: Changing message attributes

PIPE NETV LIST DSILog | COLOR BLUE REV | CONSOLE

```
NetView V6R1M0          Tivoli NetView   AOFDA NETOP1  06/13/11 19:47:27
* AOFDA    PIPE NETV LIST DSILog | COLOR BLUE REV | CONSOLE
- AOFDA    TYPE: OPT TASKID: DSILog  TASKNAME: DSILog  STATUS: ACTIVE
- AOFDA    MEMBER: DSILogK
- AOFDA    PRIMARY:DSILOG STATUS:ACTIVE  SECONDARY:DSILOGS STATUS:INACTIVE
- AOFDA    AUTOFLIP: YES             RESUME: YES
- AOFDA    LOADMOD: DSIZDST
- AOFDA    Task Serial: 332 REXX Environments: 1 (1%)
- AOFDA    Messages Pending: 0 Held: 0
- AOFDA    WLM Service Class: Not Available
- AOFDA    END OF STATUS DISPLAY
```

Modify the entire multiline response to display as reverse video blue

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **COLOR** (presentation attributes) stage changes how messages look at the NetView console. You can change the color, highlighting, and intensity of data records in a pipeline:

- Color: Blue, red, pink, yellow, green, white, turquoise, default
- Highlighting: Reverse, underscore, blink, none
- Intensity: Normal, bright, dark

Synonyms of COLOR are COLOUR and PRESATTR.

LOCATE stage: Selecting data from pipeline

IBM

LOCATE stage: Selecting data from pipeline

Pipe NETV List Status=Tasks | LOCATE 55.10 /NOT ACTIVE/ | CONSOLE CLEAR

NetView V6R1MO	Tivoli NetView	AOFDA NETOP1	06/13/11 19:50:48
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSITRACE	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSIELTSK	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSICORSV	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: ALIASAPL	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSIAL2WS	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSIAOPT	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSIB2MT	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSIGBS	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSIKREM	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSIROWS	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSIRTR	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: DSIMPTSK	STATUS: NOT ACTIVE
- AOFDA	TYPE: OPT TASKID:	TASKNAME: VPDTASK	STATUS: NOT ACTIVE

Locate only NetView tasks that are NOT ACTIVE (position 55, length 10)

1-52

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **LOCATE** stage selects messages that match a specified delimited character string to be passed to the primary output stream. Messages that do not contain the character string pass to the secondary output stream, if connected. If no secondary output stream exists, the messages that do not match the LOCATE are discarded.

LOCATE examines each input message for a match to the delimited string. You can supply a position and length pair to limit the search to a particular column range. An example follows:

```
'Pipe NETV List Status=Tasks',
' | LOCATE 55.10 /NOT ACTIVE/ | CONSOLE CLEAR'
```

This stage issues the NetView **LST STATUS=TASKS** command and places the output in the pipeline. Only records that contain the string NOT ACTIVE beginning in position 55 for a length of 10 are selected.

NLOCATE stage: Discarding data from pipeline



NLOCATE stage: Discarding data from pipeline

Pipe NETV List Status=Tasks | **NLOC 55.10 /NOT ACTIVE/ | OPT/ | PPT/ | MNT/**
| LOC/AUTO/ | COLOR GRE | CONSOLE CLEAR

	NetView V6RIMO	Tivoli NetView	AOFDA NETOP1	06/13/11 19:54:55
- AOFDA	TYPE: OST TASKID: AUTOAON	RESOURCE: AUTOAON	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: AUTOCCTS	RESOURCE: AUTOCCTS	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: AUTOCCT6	RESOURCE: AUTOCCT6	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: AUTOCCT7	RESOURCE: AUTOCCT7	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: AUTODC1	RESOURCE: AUTODC1	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: AUTODC2	RESOURCE: AUTODC2	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: AUTODC3	RESOURCE: AUTODC3	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: AUTODC4	RESOURCE: AUTODC4	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: AUTOEDAT	RESOURCE: AUTOEDAT	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: AUTONALC	RESOURCE: AUTONALC	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: AUTONVSP	RESOURCE: AUTONVSP	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: AUTOPSAV	RESOURCE: AUTOPSAV	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: AUTOTMSI	RESOURCE: AUTOTMSI	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: AUTOXCF	RESOURCE: AUTOXCF	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: AUTOXDSC	RESOURCE: AUTOXDSC	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: AUTO1	RESOURCE: AUTO1	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: DBAUTO1	RESOURCE: DBAUTO1	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: DBAUTO2	RESOURCE: DBAUTO2	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: AUTO2	RESOURCE: AUTO2	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: AUTOTCP	RESOURCE: AUTOTCP	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: AUTOPKTS	RESOURCE: AUTOPKTS	STATUS: ACTIVE	
- AOFDA	TYPE: OST TASKID: AUTOTCP	RESOURCE: AUTOTCP	STATUS: ACTIVE	

Locate subset of NetView autotasks that are ACTIVE

1-53 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **NLOCATE** stage discards messages from the primary output stream that match a specified delimited character string. Messages that do not contain the character string are passed to the primary output stream. Messages that contain the character string are passed to the secondary output stream, if connected.

You can supply a position and length to limit the search to a particular column range. An example follows:

```
'Pipe NETV List Status=Tasks',
' | NLOC 55.10 /NOT ACTIVE/ | COLOR GRE | CONSOLE CLEAR'
```

This stage issues the NetView **LST STATUS=TASKS** command and places the output in the pipeline. Only the records that do not contain the string NOT ACTIVE beginning in position 55 for a length of 10 are selected.

The example on the slide contains other data to NLOCATE. In this case, data for all tasks of type OPT, PPT, and MNT are to be discarded. The LOCATE stage further refines the pipeline data by finding tasks that begin with AUTO.

TOSTRING stage: Ending data stream if string is found



TOSTRING stage: Ending data stream if string is found

Pipe MVS D A,L | corr 10 | **tostring /IEE114I/** | cons

```

NetView V6R1M0          Tivoli NetView   AOFDA NETOPI  06/13/11 20:01:59
* AOFDA    PIPE MVS D A,L | CORR 10 | TOSTRING /IEE114I/ | CONS
" AOFDA
IEE114I 20.01.59 2011.164 ACTIVITY 528
JOBS   M/S   TS  USERS    SYSAS   INITS   ACTIVE/MAX VTAM      OAS
00003  00024  0002   00033  00013  00002/00040  00017
CANACN CANACN  CNDL   NSW   S   LLA     LLA     NSW   S
JES2    JES2    IEFPROC  NSW   S   VLF     VLF     NSW   S
VTAM   VTAM    VTAM   NSW   S   DLF     DLF     NSW   S
RACF   RACF    RACF   NSW   S   TSO     TSO     STEP1  OWT   S
SDSF   SDSF    SDSF   NSW   S   TCPPIP  TCPPIP  TCPPIP NSW   SO
TN3270 TN3270  TN3270 NSW   SO DB9GMSTR DB9GMSTR IEFPROC NSW   S
HTTPD1 HTTPD1  WEBSRV1 IN    SO NFSS   NFSS   GFSAMAIN NSW   SO
IBMSM  IBMSM   ISM13  NSW   SO DB9GIRLM DB9GIRLM NSW   S
DB9GDBM1 DB9GDBM1 IEFPROC NSW   S   DB9GDIST DB9GDIST IEFPROC NSW   SO
PORTMAP PORTMAP PMAP   OWT   SO OSNMPD  OSNMPD  OWT   SO
SNMPQE SNMPQE SNMPQE OWT   SO FTPD1  STEP1   FTPD   OWT   AO
INETD4 INETD4  STEP1   OMVKERN OWT   AO SSHD4  STEP1   START2 OWT   AO
NETVSSI NETVSSI NETVIEW NSW   S   CNMEUNIX CNMEUNIX *OMVSEX OWT   SO
NETVIEW NETVIEW  NETVIEW NSW   SO
PHOLM  OWT    IBMUSER OWT
-----
```

Issue MVS D A,L command with *terminating condition*: TOSTRING /IEE114I/

1-54 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

With the **TOSTRING** stage, you can select messages containing the text that match a specified string. Selected messages pass to the primary output stream. Those not selected pass to the secondary output stream, if connected. TOSTRING generates a *terminating condition* for the CORRWAIT stage. TOSTRING ends the preceding CORRWAIT stage.

In the example, if you do not specify a terminating condition, the IEE114I is displayed, but the pipeline waits the full 10 seconds before it disconnects and moves to the CONS stage.

You can specify up to 40 delimited strings, each with an optional position and length pair to limit the column range of the search.

TAKE stage: Selecting data based on line number

IBM

TAKE stage: Selecting data based on line number

PIPE NETV PING 10.44.15.200 | CORR 10 | **TAKE LAST 1** | CONS

C AOFDA BNH765I Pinging tived1.tived.ibm.com at 10.44.15.200 with 3 packets
of length 16 bytes
C AOFDA BNH767I 16 bytes received from 10.44.15.200: seq=1 in 0ms
C AOFDA BNH767I 16 bytes received from 10.44.15.200: seq=2 in 1ms
C AOFDA BNH767I 16 bytes received from 10.44.15.200: seq=3 in 0ms
C AOFDA BNH769I 3 packets sent, 3 packets received, 0.00% packet loss
C AOFDA BNH770I Round trip times from 0 to 1 ms, averaging 0ms

Issue NetView **PING** command. Keep only the last message, BNH770I, in output stream. All other messages are to be discarded (sent to secondary output stream)

1-55 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

With the **TAKE** stage, you can specify the number of messages or lines that pass to the primary output stream (if any). All messages or lines that exceed this number pass to the secondary output stream.

TAKE disconnects the input stream with discarded messages that pass to the secondary output stream. TAKE generates a *terminating condition* for the CORRWAIT stage. TAKE LAST *nn* waits until the last message is displayed (or a timeout occurs).

DROP stage: Discarding data based on line number

IBM

DROP stage: Discarding data based on line number

Pipe netv DISPPI ALL | corr | sep | **drop first 3 | drop last 1** | coll | cons clear

NetView V6R1MO	Tivoli NetView	AOFDA NETOP1	06/13/11 20:06:02
* AOFDA			
DW0951I NETVALRT	ACTIVE	1000	0 0043
DW0951I DSICQSK	ACTIVE	100	0 0043
DW0951I AOFDAHIM	ACTIVE	1000	0 0043
DW0952I NETVRCV	INACTIVE	2	0
DW0951I CNMUNIX	ACTIVE	1000	0 0042
DW0951I CNMPCMDR	ACTIVE	2000	0 0043

Issue NetView **DISPPI ALL** command. Discard the first three lines (header) and last line. All other messages are to be kept and collected into a single multiline message

1-56 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

With the **DROP** stage, you can specify how many messages or lines are to be discarded from the primary output stream. When the specified number of messages are read from the input stream and discarded, all other messages copy to the primary output stream. Discarded messages or lines pass to the secondary output stream, if connected.

COLLECT and SEPARATE stages: Processing messages

IBM

COLLECT and SEPARATE stages: Processing messages

Pipe netv DISPPI ALL | corr | **sep** | drop first 3 | drop last 1 | **collect** | cons

```
NetView V6R1MD      Tivoli NetView   AOFDA NETOP1  06/13/11 20:10:27
* AOFDA  PIPE NETV DISPPI ALL | CORR | SEP | DROP FIRST 3 | DROP LAST 1 |
    COLLECT | CONS

' AOFDA
DWO951I NETVALRT ACTIVE      1000      0      0      0 0043
DWO951I DSQITSK ACTIVE      100       0      0      0 0043
DWO951I AOFDAHTM ACTIVE     1000      0      0      0 0043
DWO952I NETVRCV INACTIVE     2        0      0      0 0043
DWO951I CNMUNIX ACTIVE     1000      0      3      0 0042
DWO951I CNMPCMDR ACTIVE     2000      0      0      0 0043
```

Issue NetView **DISPPI ALL** command. **SEPARATE** multiline response.
Discard first three lines (header messages) and last line. All other messages (DWO951I) are to be kept as a single multiline message with a **COLLECT** stage

1-57

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **SEPARATE** stage transforms multiline messages into multiple single-line messages. Input single-line messages pass without modification. Output single-line messages inherit all of the attributes of the input messages that created them. The output of **SEPARATE** consists of single-line messages.

The **COLLECT** stage creates a multiline message from one or more messages in the pipeline. The attributes of the output are inherited from the first line that is collected for that message.

CHOP stage: Truncating pipeline data



CHOP stage: Truncating pipeline data

Pipe NETV List Status=Tasks | LOCATE 55.10 /ACTIVE/ | LOCATE 19.3 /AUT/ | CHOP 46 | CONSOLE CLEAR

```
TYPE: OST TASKID: AUTDVIPA RESOURCE: AUTDVIPA
TYPE: OST TASKID: AUTOAON RESOURCE: AUTOAON
TYPE: OST TASKID: AUTODC1 RESOURCE: AUTODC1
TYPE: OST TASKID: AUTODC2 RESOURCE: AUTODC2
TYPE: OST TASKID: AUTODC3 RESOURCE: AUTODC3
TYPE: OST TASKID: AUTODC4 RESOURCE: AUTODC4
TYPE: OST TASKID: AUTODC5 RESOURCE: AUTODC5
...
TYPE: OST TASKID: AUTTCP9 RESOURCE: AUTTCP9
TYPE: OST TASKID: AUTTCP10 RESOURCE: AUTTCP10
TYPE: OST TASKID: AUTTRAP RESOURCE: AUTTRAP
TYPE: OST TASKID: AUTWKSTA RESOURCE: AUTWKSTA
```

Issue **LIST STATUS=TASKS** command. Locate a subset of NetView autotasks that are ACTIVE. Delete **STATUS: ACTIVE** text from each line

1-58

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **CHOP** stage truncates lines after a specified column, character, or string. The data that CHOP keeps passes to the primary output stream. The data that CHOP discards passes to the secondary output stream, if connected. In this example, all data after column 46 is deleted. This action removes **STATUS: ACTIVE** from each line in the pipeline.

BETWEEN stage: Dividing a message stream

IBM

BETWEEN stage: Dividing a message stream

```
Pipe < CNMPNL1.EUYSLIST | TAKE 90
| BETWEEN 1.14 :IF DTYP=MSG/ 1.6 :ENDIF/ | CONS ONLY
```

```
CNMKWDN OUTPUT FROM PIPE < CNMPNL1 EUYSLIST | TAKE 90 | LINE 0 OF 24
----- Top of Data -----
: IF DTYP=MSG/
  Enter HELP PIPE DSIVSAM to display syntax and general usage.
  Enter HELP DSIVSAM DEL to receive help on deleting records.
  Enter HELP DSIVSAM GET to receive help on reading VSAM records.
  Enter HELP DSIVSAM GETREV to receive help on reading VSAM records in
    reverse sequence.
  Enter HELP DSIVSAM INQUIRE to receive help on displaying data set
    characteristics for a file.
  Enter HELP DSIVSAM PUT to receive help on creating or replacing records.
:ENDIF
: IF DTYP=MSG/
  Enter HELP PIPE DSIVSMX to display syntax and general usage notes.
  Enter HELP DSIVSMX CLOSE to receive help on closing records.
  Enter HELP DSIVSMX DEL to receive help on deleting records.
  Enter HELP DSIVSMX GET to receive help on reading records.
  Enter HELP DSIVSMX GETREV to receive help on reading records in
    reverse sequence.
  Enter HELP DSIVSMX IDCAMS to receive help on data set maintenance.
  Enter HELP DSIVSMX INQUIRE to receive help on displaying data set
    characteristics for a file.
  Enter HELP DSIVSMX OPEN to receive help on opening files locally to the
    issuing task.
  Enter HELP DSIVSMX PUT to receive help on putting records.
:ENDIF
----- Bottom of Data -----*
```

Read member EUYSLIST. Keep all data between :IF DTYP=MSG/ and :ENDIF statements

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **BETWEEN** stage divides a message stream into sections. The selected sections begin with a message containing a specified string and end with either a specified string or a number of messages. The selected sections pass to the primary output stream. The sections that were not selected pass to the secondary output stream. All lines in EUYSLIST between the :IF DTYP=MSG/ and :ENDIF statements are included in the pipeline data.

REVERSE stage: Reversing order of pipeline data



REVERSE stage: Reversing order of pipeline data

Pipe MVS D A,L | CORR | take first 1 | reverse message | cons clear

```
" AOFDA
ITEMUSER OWT
CANSNA CANSNA AGENT NSW SO
OSNMPD OSNMPD OWT SO CANSDSST CANSDSST TEMS NSW SO
SYSLOGD8 STEP1 OMVSKERN NSW AO RXSERVE RXSERVE RXSERVE OWT SO
TCP IP TCP IP TCPIP NSW SO INETD4 STEP1 OMVSKERN OWT AO
RACF RACF RACF NSW S TSO TSO TSO STEP OWT S
AUTOSSI AUTOSSI NETVIEW NSW S AUTONETV AUTONETV NETVIEW NSW SO
RMF RMF IEFPROC NSW S RRS RRS RRS NSW S
JES2 JES2 IEFPROC NSW S VTAM VTAM VTAM NSW S
LLA LLA LLA NSW S VLF VLF VLF NSW S
00002 00015 00001 00028 00007 00001/00300 00009
JOBS M/S TS USERS SYSAS INITS ACTIVE/MAX VTAM OAS
IEE114I 15.19.30 2007.249 ACTIVITY 425
```

Issue MVS D A,L command. Reverse output order. No need to separate message

1-60

IBM Software Group | Tivoli Software
 © 2011 IBM Corp.

You can use **REVERSE** stage to reverse the order of data in the pipeline as follows:

- LINE (default): Specifies that each line of output is to be reversed, character by character. For example, DSI069I SYNTAX ERROR changes into RORRE XATNYS I960ISD.
- MESSAGE: Specifies that each multiline message is to be reversed, line by line. Affects only multiline messages.
- STREAM: Specifies that the next stage receives messages in reverse order. This option has no effect on the structure of multiline messages.
- REVERSE waits until all messages are in the pipeline before processing the data.

PICK stage: Selecting specific data

IBM

PICK stage: Selecting specific data

```
PIPE NETV TASKUTIL | SEP | DROP FIRST 3 | DROP LAST 5
| PICK 20.7 > / 11 | COLL | CONS
```

	NetView V6R1MO	Tivoli NetView	AOFDA NETOPI	06/13/11 20:18:53
*	AOFDA	PIPE NETV TASKUTII SEP DROP FIRST 3 DROP LAST 5 PICK 20.7 >		
	/	1/ COLL CONS		
	AOFDA			
	AOFDAPPT PPT 255	84.51 0.00 0.00 0	577 **NONE**	
	AUTOAON AUTO 250	44.68 0.00 0.00 0	548 **NONE**	
	AUTO1 AUTO 250	13.54 0.00 0.00 0	332 **NONE**	
	AUTO2 AUTO 250	13.62 0.00 0.00 0	272 **NONE**	

CPU time, in seconds

Issue **TASKUTIL** command and select only the lines for tasks that have greater than 1 second of CPU time (column 20, length of 7)

1-61

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **PICK** stage selects messages satisfying a criteria and passes them to the primary output stream. Messages that do not meet the specified criteria pass to the secondary output stream, if connected.

The **PICK** stage examines only the first line of a multiline message. If selected, the entire multiline message passes to the primary output stream. Use the SEParate stage to break a multiline message into multiple single-line messages.

Specify the selection criteria by giving a *position.length* within the message line to be compared against: another *position.length* within the message line or a character string. In this example, the TASKUTIL output is interrogated for any task that has consumed more than 1 second of CPU time. Because the field being checked has a length of seven, the string that is specified on PICK must contain seven characters between the delimiters. The delimiters are forward slashes.

SUBSYM stage: Substituting symbolic variables



SUBSYM stage: substituting symbolic variables

```
NETVASIS Pipe LIT /System name =  
  &SYSNAME/  
  | LIT /NetView Domain =  
    &DOMAIN/  
  | LIT /TCP name = &CNMTCPN/  
  | LIT /Network ID = &CNMNETID/  
  | SUBSYM | CONS CLEAR
```

```
NetView V6R1MO          Tivoli NetView   AOFDA NETOP1  06/14/11 13:58:39  
| AOFDA     Network ID=ADCD  
| AOFDA     TCP name = TCPIP  
| AOFDA     NetView Domain = AOFDA  
| AOFDA     System name = ADCD1
```

Insert four literal strings in the pipeline. Use **SUBSYM** to display value of &sysname, &domain, &cnmnetid, and &cnmtcpn symbolic variables

1-62

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **SUBSYM** stage substitutes system or user-defined symbolic variables (including NetView &DOMAIN) into messages in the pipeline. In this case, NETVASIS is used for retaining the mixed case of the literal strings also. To preserve the order of the strings, you can insert a REVERSE STREAM before the CONSOLE stage.

CHANGE stage: Replacing string data in pipeline

IBM

CHANGE stage: Replacing string data in pipeline

Pipe netv LIST " | change /NETOP1/ /NONAME/ | cons

```
NetView V6R1MO      Tivoli NetView   AOFDA NETOP1 06/14/11 14:06:34
* AOFDA PIPE NETV LIST '' | CHANGE /NETOP1/ /NONAME/ | CONS
- AOFDA STATION: NONAME TERM: SCOTCP19
- AOFDA HCOPY: NOT ACTIVE PROFILE: DS1PROFB
- AOFDA STATUS: ACTIVE IDLE MINUTES: 0
- AOFDA ATTENDED: YES CURRENT COMMAND: PIPE
- AOFDA AUTHRCVR: YES CONTROL: GLOBAL
- AOFDA NGMFADMN: YES DEFAULT MVS CONSOLE NAME: NET1POAD
- AOFDA NGMFVSPN: NNNNN (NO SPAN CHECKING ON NMC VIEWS)
- AOFDA NGMFCMDS: YES AUTOTASK: NO
- AOFDA IP ADDRESS: N/A
- AOFDA OP CLASS LIST: NONE
- AOFDA DOMAIN LIST: AOFDA (I) AOFDB (I)
- AOFDA ACTIVE SPAN LIST: NONE
- AOFDA Task Serial: 14649 REXX Environments: 2 (1%)
- AOFDA Messages Pending: 0 Held: 0
- AOFDA WLM Service Class: Not Available
- AOFDA END OF STATUS DISPLAY
```

Issue **LIST "** command and change the operator name from NETOP1 to NONAME

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **CHANGE** stage replaces occurrences of the first specified string with the second string. Either string can be null. If the first string is null, the second string inserts at the beginning of each line. If the second string is null, all occurrences of the first one are deleted. Data between substitutions are copied without change.

If a secondary output stream is defined, changed messages pass to the primary output stream. Unchanged messages (because first string is not found) pass to the secondary output stream.

In this example, operator NETOP1 issues the **LIST "** command. Notice the **STATION: NONAME** in the first line of the response.

CASEI stage: comparing strings

IBM

CASEI stage: Comparing strings

```
NETVASIC PIPE LITERAL /ABcdefghi/
| LITERAL /abc/
| LITERAL /xyz/
| CASEI LOCATE /abC/
| CONSOLE
```

```
NetView V6R1M0          Tivoli NetView    AOFDA NETOP1   06/14/11 14:13:00
* AOFDA      PIPE LITERAL /ABcdefghi/ | LITERAL /abc/ | LITERAL /xyz/ | CASEI
|           LOCATE /abC/ | CONS
| AOFDA     abc
| AOFDA     ABcdefghi
```

Display strings that contain the characters ABC regardless of case

1-64

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **CASEI** stage compares character strings without case sensitivity to uppercase or lowercase EBCDIC characters. In this example, CASEI LOCATE /abC/ matches *ABcdefghi* and *abc*.

SPLIT stage: Dividing data based on string

IBM

SPLIT stage: Dividing data based on string

```
PIPE LITERAL /BUY IT, YOU'LL SPLIT AND LIKE IT BETTER/
| SPLIT 3 AFTER STRING /IT/
| CONSOLE

NetView V6R1M0          Tivoli NetView   AOFDA NETOP1  06/14/11 14:20:18
* AOFDA    PIPE LITERAL /BUY IT, YOU'LL SPLIT AND LIKE IT BETTER/ | SPLIT 3
|          AFTER STRING /IT/ | CONSOLE
| AOFDA    BUY IT, Y
| AOFDA    + OU'LL SPLIT AN
| AOFDA    + D LIKE IT BE
| AOFDA    + TTER
```

Divide the text string, BUY IT, YOU'LL SPLIT AND LIKE IT BETTER,
three characters after each occurrence of the string IT

1-65

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **SPLIT** stage divides a line of text into multiple lines, based on a specified string. The default delimiter is a blank. The **SPLIT 3 AFTER STRING /IT/** example displays the following text:

```
BUY IT, Y
OU'LL SPLIT AN
D LIKE IT BE
TTER
```

STRIP stage: Removing characters from a line

IBM

STRIP stage: Removing characters from a line

Pipe < LOGTBL | **strip trailing not // 8** | coll | cons

```
* LOGTBL: Lab Auto Tbl
* DSI547I DSILOG : SECONDARY VSAM DATA SET IS NOW ACTIVE
* DSI546I DSILOG : PRIMARY VSAM DATA SET IS NOW ACTIVE
*
*
if msgid='DSI546I' | msgid='DSI547I' & token(2) = 'DSILOG'
& token(2) = 'DSILOG' & token(4) = PriOrSec then
  EXEC(CMD('LOGAUTO ' PriOrSec) ROUTE(ONE *)) HOLD(Y);
```

Use **PIPE <** to read an automation table (for example, LOGTBL). Use **STRIP** to remove the sequence numbers in columns 73 to 80

1-66

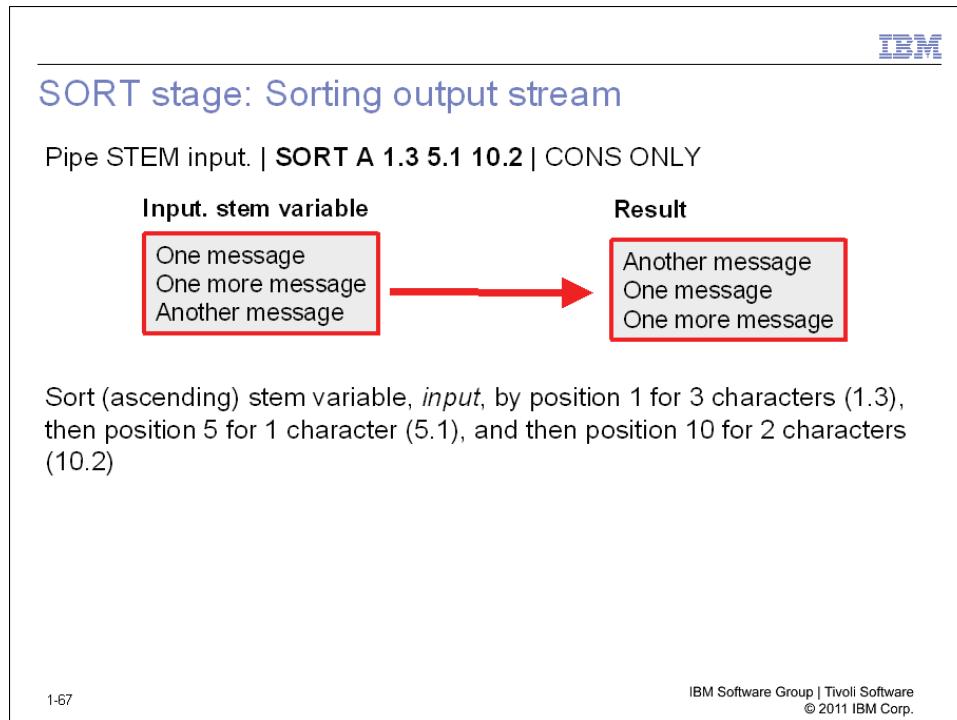
IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **STRIP** stage removes blanks or other specified characters from the beginning or end of message data. Alternately, STRIP removes all characters up to a blank or other specified characters.



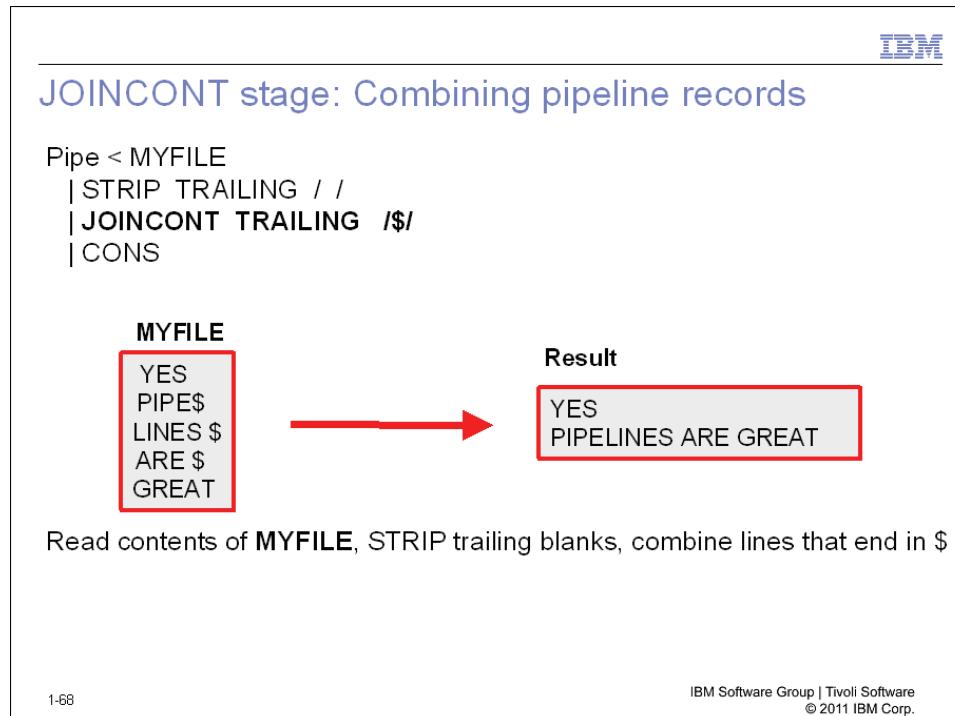
Tip: STRIP can remove unwanted blanks or other characters before you use the JOINCONT stage. Discussion about JOINCONT occurs later.

SORT stage: Sorting output stream



The **SORT** stage reads messages from the input stream and writes them to the output stream in a specified order, ascending or descending. Only the first line of each multiline message is examined. To sort lines within a multiline message, the **SEPARATE** stage must be included before **SORT**. If messages contain identical sort fields, they retain their input stream order when passed to the output stream.

JOINCONT stage: Combining pipeline records



The **JOINCONT** stage joins consecutive messages in the pipeline when a match to a specified string is found. A message is considered in its entirety, and it can include blanks or even sequence numbers if reading from a file. In this example, the pipeline records that contain a \$ are joined. **MYFILE** contains five lines. The result contains two lines.

DUPLICAT stage: Duplicating pipeline records

IBM

DUPLICAT stage: Duplicating pipeline records

Pipe LIT /MSG NETOP1,HELLO/ | DUP 3 | CORRCMD /AUTO1:
| COLOR BLUE REV | CONS

```
tView V6R1MO          Tivoli NetView   AOFDA NETOP1  06/14/11 14:33:12
AOFDA    PIPE LIT /MSG NETOP1,HELLO/ | DUP 3 | CORRCMD /AUTO1: | COLOR BLUE
         REV | CONS
AOFDA    DS1039I MSG FROM AUTO1 : HELLO
AOFDA    DS1001I MESSAGE SENT TO NETOP1
AOFDA    DS1039I MSG FROM AUTO1 : HELLO
AOFDA    DS1001I MESSAGE SENT TO NETOP1
AOFDA    DS1039I MSG FROM AUTO1 : HELLO
AOFDA    DS1001I MESSAGE SENT TO NETOP1
AOFDA    DS1039I MSG FROM AUTO1 : HELLO
AOFDA    DS1001I MESSAGE SENT TO NETOP1
```

Run the **MSG** command four times on task AUTO1. Output returns to originating operator

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **DUPLICAT** stage takes the messages in the input stream, copies them, and writes the copied messages to the output stream. The copies are marked as *copy* rather than *primary*. The message text does not change. In this example, the output from the pipeline is the DS1001I message. The DS1039I message results from the MSG NETOP1,HELLO command.

DELDUPES stage: Removing duplicate records

IBM

DELDUPES stage: Removing duplicate records

Pipe < LOGTIMES | SORT 1.18
| DELDUPES KEEPLAST 1.18 | CONS

LOGTIMES		
DOE,	JOHN	98/02/18 13:25:04
SMITH,	FRED	98/02/18 13:29:21
COLLINS,	MARY	98/02/23 17:01:55
DOE,	JOHN	98/02/23 09:00:00
HOWE,	TOM	98/02/23 04:14:20
JONES,	FRED	98/02/23 11:16:44
COLLINS,	MARY	98/03/01 10:15:40

Output stream

COLLINS, MARY 98/03/01 10:15:40
DOE, JOHN 98/02/23 09:00:00
HOWE, TOM 98/02/23 04:14:20
JONES, FRED 98/02/23 11:16:44
SMITH, FRED 98/02/18 13:29:21

Read LOGTIMES file. Delete duplicate records, keeping the last duplicate each time

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

The **DELDUPES** stage compares the first line of consecutive messages and deletes consecutive duplicates. The duplicate messages are written to the secondary output stream, if connected. To delete duplicate lines within a multiline message, use the **SEPARATE** stage before the **DELDUPES** stage.

In this example, duplicate records are removed from the LOGTIMES file based on columns 1 to 18 (last name). The first and third records are removed from the output stream. The LOGTIMES file does not change.

COUNT stage: Counting numbers of records

IBM

COUNT stage: Counting numbers of records

Pipe NETV CDRMS | CONS | COUNT **LINES FROM 0**
| COLOR BLUE REV | CONS

```
NetView V6R100          Tivoli NetView    AOFDA NETOP1  06/14/11 14:50:09
* AOFDA PIPE NETV CDRMS | CONS | COUNT LINES FROM 0 | COLOR BLUE REV | CONS
C AOFDA DISPLAY NET,CDRMS,SCOPE=ALL
A AOFDA IST097I DISPLAY ACCEPTED
. AOFDA
IST350I DISPLAY TYPE = CDRMS
IST089I CDRMS TYPE = CDRM SEGMENT
IST482I ADCD6 NEVAC, SA      6, EL  1, NETID = ADCD
IST482I ADCD7 NEVAC, SA      7, EL  1, NETID = ADCD
IST1454I END      2 RESOURCE(S) DISPLAYED
IST314I END
AOFDA 8
```

Pipe NETV CDRMS | CONS | COUNT **Messages FROM 0**
| COLOR BLUE REV | CONS

```
NetView V6R100          Tivoli NetView    AOFDA NETOP1  06/14/11 14:53:24
* AOFDA PIPE NETV CDRMS | CONS | COUNT MESSAGES FROM 0 | COLOR BLUE REV | CONS
C AOFDA DISPLAY NET,CDRMS,SCOPE=ALL
A AOFDA IST097I DISPLAY ACCEPTED
. AOFDA
IST350I DISPLAY TYPE = CDRMS
IST089I CDRMS TYPE = CDRM SEGMENT
IST482I ADCD6 NEVAC, SA      6, EL  1, NETID = ADCD
IST482I ADCD7 NEVAC, SA      7, EL  1, NETID = ADCD
IST1454I END      2 RESOURCE(S) DISPLAYED
IST314I END
AOFDA 8
```

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **COUNT** stage counts the number of messages, lines, or bytes received on its primary input stream. The count passes to its primary output stream when the input stream disconnects. The first example counts the number of lines on the pipeline (eight). The second example counts the number of messages in the pipeline (three): the **DISPLAY** command, IST097I, and IST350I. The first **CONS** stage is used for these examples to show the contents of the pipeline.



Student exercise



1-72

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 5.

Lesson 6: Processing pipeline data



Lesson 6: Processing pipeline data

- Discarding pipeline data
- Logging pipeline data
- Saving pipeline data
 - SAFE
 - KEEP

1-73

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Pipeline data can be discarded, displayed, logged, or saved in NetView storage for use later by the same EXEC or a different EXEC. The CONSOLE stage can be used for displaying data to a NetView operator.

HOLE stage: Deleting all data from pipeline



HOLE stage: Deleting all data from pipeline

- Store output of LIST command in stem variable, displaying only text message:

```
/* REXX EXEC */  
'Pipe NETV List Status=Tasks',  
'| STEM MYVAR. | HOLE',  
'| Lit / COMMAND COMPLETE / | CONSOLE'
```

- Test if command generates correlated output:

```
PIPE NETV your_command_name | HOLE
```

If you see output, *your_command_name* is not correlatable

1.74

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **HOLE** stage discards the contents of the pipeline. In the first example, HOLE is used for discarding the data in the pipeline after the data is saved to the stem variable. The LIT stage inserts text into the pipeline to say COMMAND COMPLETE. The second example can be used for determining if a command is *correlatable*, that it has correlated output. If the command response is displayed, it is not correlatable. You can also use the HOLE stage to discard output stream from a multiple-output pipeline.

LOGTO stage: Log messages

IBM

LOGTO stage: Log messages

```
-- *-----+
|--LOGTO--+-----+-----+
 +-- ALL-----+
 |-----+-----+-----+
 '- HCYLOG-' '- NETLOG-' '- SYSLOG-'
```

- LOGto: Send a copy of the pipeline contents to a specified log
The contents remain in pipeline for processing by next stage
- NETLOG: Send a copy of message to NetView DSILog
- HCYLOG: Send a copy of message to hardcopy log device
- SYSLOG: Send a copy of message to system log

```
Pipe LIT /log this message/ | LOG NETLOG
```

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

The **LOGTO** stage sends a copy of the contents of the pipeline to one or more of the specified logs. The input stream (unmodified) passes to the output stream for processing by the next **LOGTO *** stage. Pipeline contents are be logged based on default settings that are defined in CNMSTYLE or with the NetView OVERRIDE and DEFAULT commands.

The default logging settings that you can find in CNMSTYLE are as follows:

```
DEFAULTS.NetLog = Yes
DEFAULTS.SysLog = No
DEFAULTS.HcyLog = Yes
```

You can issue a **LIST DEFAULTS** command to display the settings as follows:

```
DWO654I DISPLAY      DEFAULTS
SYSLOG: NO
NETLOG: YES
HCYLOG: YES
And so on
```

SAFE stage: Storing pipeline data

SAFE stage: Storing pipeline data

```
.- *-----  
>>-SAFE-+-----+-----+-----><  
'- name-'  +- APPEND+-  
'- SEIZE--'
```

SAFE: A place to store one or more messages associated with a procedure, retaining all attributes of the messages

- Procedures can read from and write to the safe place
When writing, the data can be replaced or appended
- Default safe (*) place holds only one message, usually message from automation
- Named safe place can hold multiple messages
Can also be used by nested procedures
- SEIZE reads messages from place and empties it, improving performance
- The safe place empties when procedure ends

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

The **SAFE** stage defines a place to store one or more messages *associated with a command procedure*, for example, a REXX EXEC. With the SAFE stage, you can read from or write to a default or a named SAFE.



Note: The messages in a SAFE retain their full message structure and attributes.

When SAFE is the first stage, the specified SAFE is read into the pipeline. For a named SAFE, multiple messages could potentially go into the pipeline.

When SAFE is not the first stage, the input messages are written to the specified SAFE. For the default SAFE, just one message is written and all messages are copied to the output stream. For a named SAFE, each input message is written to the SAFE and to the output stream.

Default SAFE

The default SAFE is the current message that is associated with a command procedure. For example, when the automation table invokes a command procedure, the default safe contains the automated message. The default SAFE is preserved while the command procedure is active. The default SAFE can contain only one message.

Named SAFE

The named SAFE is a named area for a queue of messages associated with a command procedure or group of nested command procedures. For example, a REXX EXEC can write messages to a named SAFE, then call a second EXEC to read from, or write to, the named SAFE.



Note: A named SAFE is preserved while the command procedure or command group is active.

A named SAFE can contain any number of messages. A command procedure group can have any number of named SAFEs at a time.

- APPEND: Specifies that data should be added after data that already exists in the named SAFE. APPEND is valid only when using a named SAFE.
- SEIZE: Use SEIZE for performance improvement when you do not need the contents of the safe to remain in the safe after a read operation.

SAFE stage examples



SAFE stage examples

- Save data in named safe, MYSAFE, for use twice in procedure
 - Save MYSAFE data in a variable
 - Display MYSAFE data on screen

```
/* REXX EXEC */  
'Pipe LIT /Save This/ |  SAFE MYSAFE'  
. . .  
'Pipe SAFE MYSAFE | VAR text_string'  
'Pipe SAFE MYSAFE | CONSOLE'
```
- Read and empty contents of named safe storing data in common global array for use by other procedures

```
'Pipe SAFE SYSERR SEIZE | STEM (COMMON)  SYSERR. '
```

1.77

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The second SAFE example reads the contents of the named safe, SYSERR, into the common global (stem) variable *syserr*, emptying the contents of the safe.

KEEP stage

KEEP stage

```
>>-KEEP-- keepname---+-----+-----+----->
      +- timeout-+  +- APPEND-+
      '- *-----'  '- SEIZE--'

>-+- NOSPILL-----+-----><
  +- SPILL-----+
  '- ENDCMD /cmd_string/-'
```

KEEP: Defines a task-wide place to store messages, retaining all attributes of the messages

- Procedures can read from and write to the keep place, replacing or appending data
- If no data is written to keep before *timeout* expires, the keep place is purged
Adding data to the keep resets the interval
- SPILL displays contents of the keep place when *timeout* expires
- ENDCMD defines a command to be issued for each message in the keep place when *timeout* expires
- SEIZE reads messages from the keep place and empties it, improving performance

1-78 IBM Software Group | Tivoli Software
© 2011 IBM Corp.

The **KEEP** stage is similar to the **SAFE** stage. With the **KEEP** stage, you can define a task-wide place to store messages. You can read from or write to storage that the *keepname* defines. The name and message are global for the task and exist beyond the life of the procedure that creates them.

If PIPE KEEP is the first stage, it copies messages from the **KEEP** into the output stream. If PIPE **KEEP** is *not* the first stage, it copies messages from the input stream into the **KEEP**.

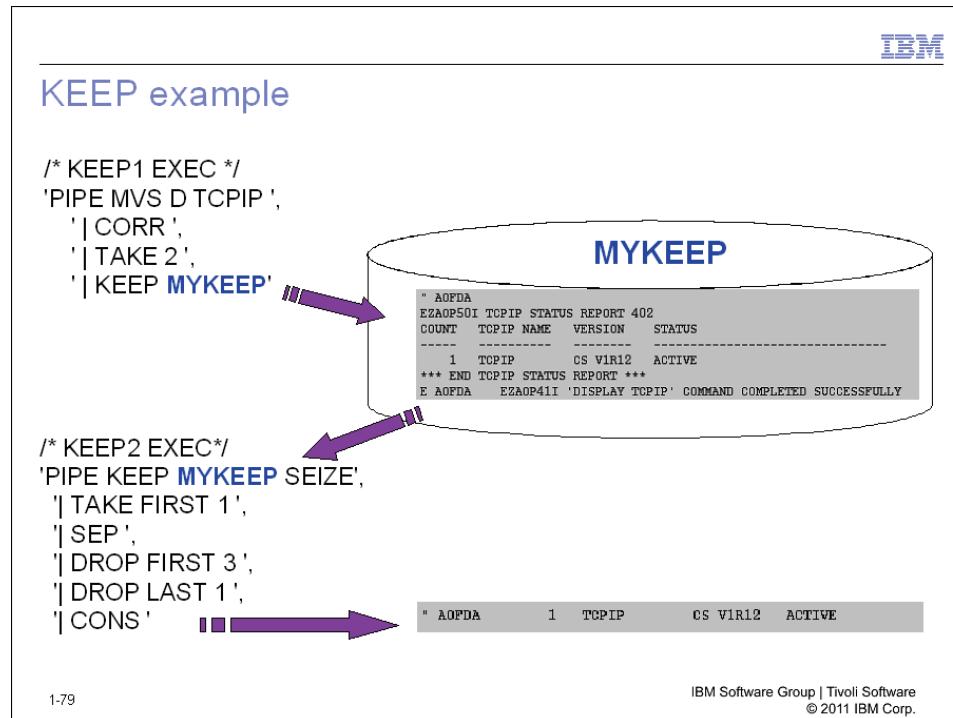
You can use the **SPILL** operand when **KEEP** is not a first stage. When the **KEEP** expires, **SPILL** indicates to display messages that are in the **KEEP**. The messages become subject to automation and message traps. If the **KEEP** expires because of a LOGOFF command or task termination, messages route to the authorized receiver.

You can use the **NOSPILL** operand when **KEEP** is not a first stage. When the **KEEP** expires, **NOSPILL** indicates to discard messages that are in the **KEEP**.

Use the **QRYKEEP** command to display keep statistics as follows:

```
BNH560I KEEP Status for NETOP1
Keep Name      Number Messages    Total Storage     Time Out
KEEP1          40                  18212             1000
KEEP2          1                   1040              *
```

KEEP example



This slide shows a simple example of using a KEEP to share data between two REXX EXECS:

1. KEEP1 is a REXX EXEC that issues the MVS D TCPIP command and stores the output in a KEEP called MYKEEP. Immediately after storing the data, KEEP1 ends.
2. EXEC KEEP2 runs. KEEP2 reads the contents of MYKEEP and displays one message on the NetView console. As KEEP2 reads MYKEEP, it deletes the data from the keep with the SEIZE option.

Student exercise

Student exercise



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 6.

Lesson 7: Advanced topics



Lesson 7: Advanced topics

- Cross-domain pipelines
- Pipe within pipe
- Multiple input and output streams
- Exposing pipeline messages to automation
- Building large pipelines dynamically: INTERPRT stage

1-81

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This lesson discusses advanced pipeline techniques. The INTERPRT stage is similar to a REXX INTERPRT instruction. Using the INTERPRT stage, you can dynamically build and run more complex pipelines.

Cross-domain pipelines

IBM

Cross-domain pipelines

Pipe CORRCMD AOFDB: list dsilog | color blue | lit /Cross-domain PIPE to AOFDB:/ | CONS

```
NetView V6R1M0          Tivoli NetView     AOFDA NETOP1   06/14/11 16:09:04
* AOFDA      PIPE CORRCMD AOFDB: LIST DSILog | COLOR BLUE | LIT /CROSS-DOMAIN
|           PIPE TO AOFDB:/ | CONS
| AOFDA      CROSS-DOMAIN PIPE TO AOFDB:
- AOFDB
TYPE: OPT TASKID: DSILog    TASKNAME: DSILog    STATUS: ACTIVE
MEMBER: DSILogBK
PRIMARY:DSILogP  STATUS:ACTIVE    SECONDARY:DSILOGS  STATUS:INACTIVE
AUTOFILIP: YES             RESUME: YES
LOADMOD: DSIZDST
Task Serial: 321  REXX Environments: 1 (1%)
Messages Pending: 0 Held: 0
WLM Service Class: Not Available
END OF STATUS DISPLAY
```

Using a labeled RMTCMD: Issue a **LIST DSILog** command in remote domain AOFDB on same operator ID in as AOFDA (NETOP1). Return results in blue as a multiline message. A literal string is also inserted

1-82

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This example shows a labeled RMTCMD using CORRCMD to issue a LIST DSILog in a remote NetView domain. The PIPE is issued from NETOP1 in the AOFDA domain to the AOFDB domain as the same operator. The LIST DSILog response returns as blue text. The LITERAL string remains unchanged because it was inserted after the COLOR stage. The CORRCMD stage includes any pertinent pipeline stages from DSICCDEF.

Pipe within a pipe

Pipe within a pipe

1 Using a labeled RMTCMD: Issue a **PIPE** command to **PING** a resource

PIPE CC AOFDB:

```
PIPE (STAGESEP %) NETV PING 10.44.15.200 % CORR 3
% TAKE LAST 2 % COLOR BLUE REV % CONS ONLY | CONS ONLY
```

NetView V6RIMO	Tivoli NetView	AOFDA NETOP1	06/14/11 16:14:40
* AOFDA	PIPE CC AOFDB:	PIPE (STAGESEP %) NETV PING 10.1.32.49 % CORR 3 %	
		TAKE LAST 2 %COLOR BLUE REV % CONS ONLY CONS ONLY	
C AOFDB	BNH769I	3 packets sent, 3 packets received, 0.00% packet loss	
C AOFDB	BNH770I	Round trip times from 3 to 7 ms, averaging 4ms	



2 Take the last 2 lines in the response and return results in reverse-video blue to the outer pipeline

The **PING** command runs in remote domain AOFDB under the same operator ID as in AOFDA (NETOP1)

1-83

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

This example shows how to issue a *pipe within a pipe* for sending a command to a cross-domain NetView. A *pipe within a pipe* produces performance while a second system processes data and returns only necessary responses.

A second stage separator character (%) is necessary to distinguish pipeline stages between the inner and outer pipes. Any CORRWAIT value in the inner pipe must be smaller than the CORRWAIT value in the outer pipe.

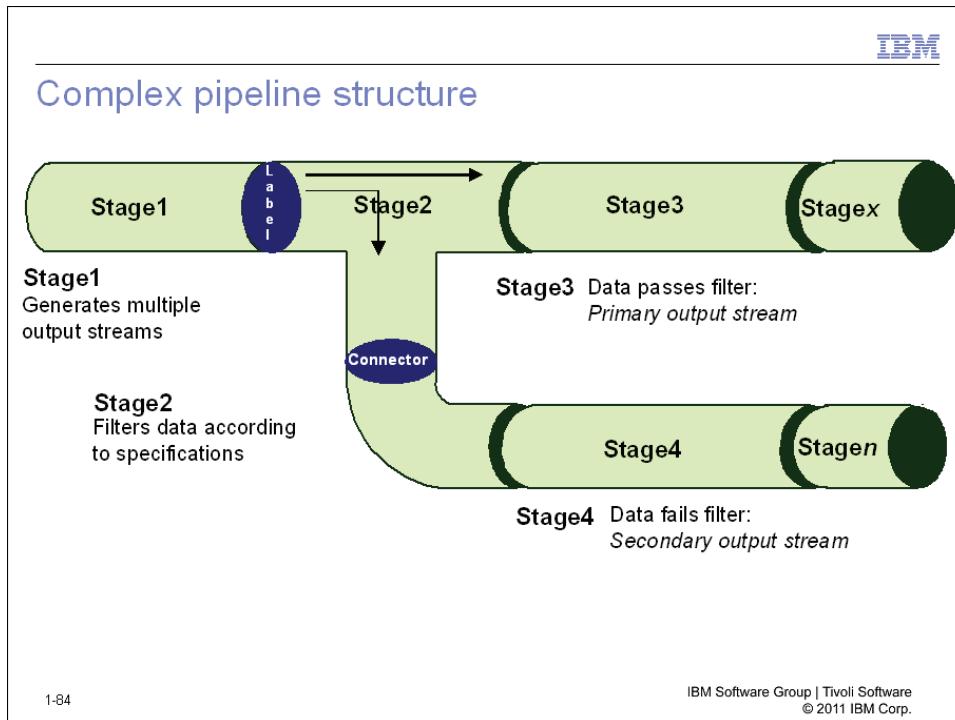
The complete ping response is as follows:

```
BNH765I Pinging tived1.tived.ibm.com at 10.1.32.49 with 3 packets
of length 16 bytes
BNH767I 16 bytes received from 10.1.32.49: seq=1 in 3ms
BNH767I 16 bytes received from 10.1.32.49: seq=2 in 7ms
BNH767I 16 bytes received from 10.1.32.49: seq=3 in 3ms
BNH769I 3 packets sent, 3 packets received, 0.00% packet loss
BNH770I Round trip times from 3 to 7 ms, averaging 4ms
```

This pipeline uses only the last two messages.

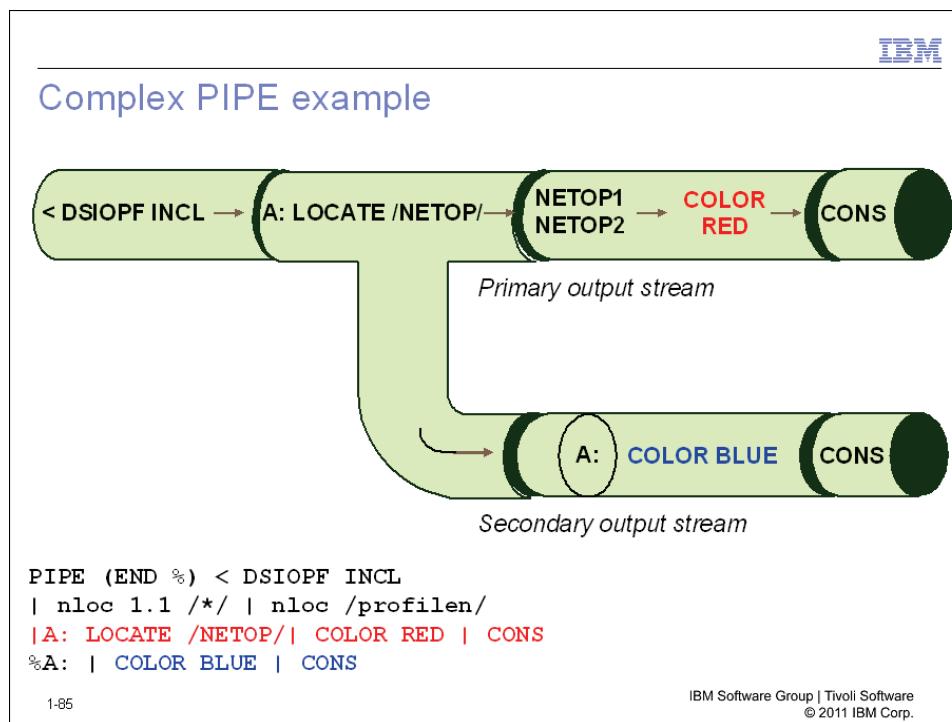
The inner pipeline ends with a CONS ONLY stage. The messages in the output stream flow to the input for the next stage of the outer pipeline, another CONS ONLY stage. CONS ONLY is necessary for both inner and outer pipelines to display the response to the PING command.

Complex pipeline structure



A complex pipeline is made up of several simple pipelines that are connected with labels. Complex pipelines are simple programs rather than complicated commands.

Complex PIPE example



This example reads all lines that are contained in the DSIOPF file (plus included members) into the pipeline. The label **A:** on the LOCATE stage creates a *secondary* output stream for the LOCATE stage. All data in the pipeline that matches the LOCATE stage specification (inclusion of the NETOP character string) passes to the *primary* output stream. All data that does not match passes to the secondary output stream.

The COLOR stage modifies the color attribute of the records that the LOCATE stage selects to red on its primary output stream.

A connector labeled **%A** indicates the end of the first (simple) pipeline that is identified by the label **A:**. It indicates the target of the secondary output stage of the first pipeline LOCATE stage. All data in the secondary output stream is to change its color attribute to blue.

NOT stage: Exchanging input and output streams

IBM

NOT stage: Exchanging input and output streams

Pipe MVS D T | CORR | TAKE 1 | NOT CHOP 8 | CONSOLE

```
NetView V6R1M0      Tivoli NetView    AOFDA NETOP1   06/14/11 16:37:20
* AOFDA    PIPE MVS D T | CORR | TAKE 1 | NOT CHOP 8 | CONSOLE
E AOFDA    LOCAL: TIME=16.37.19 DATE=2011.165 UTC: TIME=15.37.19 DATE=2011.165
```

- The command deletes the first eight characters for each line of the MVS D T command output
- Secondary output stream contains the first eight characters, IEE136I, and a blank

1-86

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

You can use the **NOT** stage to exchange the input and output streams. CHOP, TAKE, TOSTRING, and LOCATE are some of the stages that can use a NOT stage.

Multiple input and output streams



Multiple input and output streams

- PIPE stages can have multiple input and output streams:
 - FANOUT: Up to ten output streams are supportable
 - FANIN and FANINANY: Up to ten input streams are supportable
- Adjacent stages:
Output stream of first stage becomes input stream of second stage
- Non-adjacent stages:
 - Using label (connector) passes output as input
 - Using END character identifies end of one simple pipeline and start of the next pipeline
- Related PIPE stages:
 - LOOKUP stage compares two input streams
 - PIPEND stage causes a complex pipeline to end immediately

1-87

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

When stages are adjacent each other in a pipeline, the output stream of a stage connects to the input stream of the stage that follows. When stages are *not* adjacent each other in a pipeline, labels and connectors are used for defining the input stream to the non-adjacent stages.

The *end* character indicates the end of one pipeline and the beginning of the next pipeline. The % symbol (END %) is the most common character.

A label is used for creating multiple data streams for a stage. These data streams are numbered, starting with one, and can go as high as 10, depending on the stage. When a stage is processed, the number one (primary) input stream connects to the previous stage if there is one. The number one (primary) output stream connects to the following stage if there is one.

When a connector is encountered later in the pipeline specification, a data stream is defined. The data stream connects from the stage where the label was defined to the connector. The lowest stream number available is assigned to the data stream.

If the labeled stage has an output in a simple pipeline within a complex pipeline, the data stream becomes an output from the stage that defines the label. It also becomes an input to the stage that follows the connector. If the labeled stage is an output to a stage in the pipeline specification, the data stream becomes an input to the stage that defines the label. The data stream also becomes an output to the stage that precedes the connector.

It is possible for a connector to be neither first nor last, in which case, the connector defines both an input and an output for the labeled stage. It is also possible to use two connectors in a row. This usage connects the output of one labeled stage to the input of another.

FANIN, FANINANY, and FANOUT

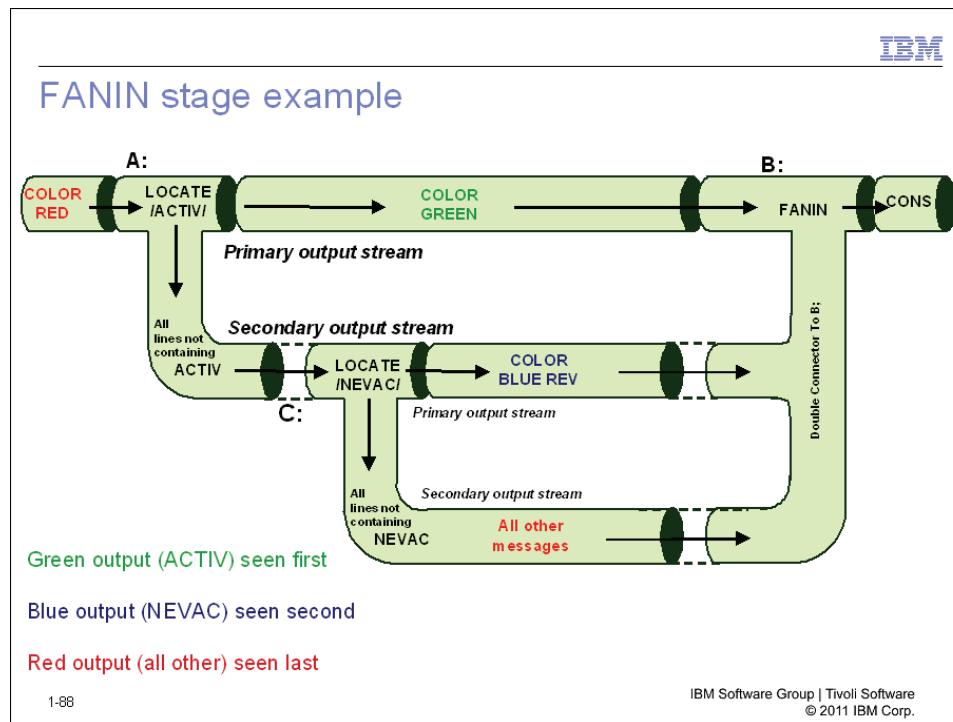
The FANIN stage reads from multiple input streams. FANIN reads from the first stream until that stream disconnects. FANIN then reads from the next input stream until it disconnects, and so on. All data that FANIN reads passes to a single output stream.

The FANINANY stage reads from each connected input stream and passes the messages to a single output stream. Messages pass in the order received regardless of their input stream.

The FANOUT stage copies the messages that it receives on its primary input stream to all connected output streams. Messages that copy to the output streams are identical except for the following cases:

- Messages that are written to the primary output stream retain the IFRAUPRI primary attribute of the original message. The copy attribute, defined by IFRAUCPY, is set to zero.
- Messages that are written to all other output streams have the IFRAUPRI primary attribute setting set to zero and the copy attribute, defined by IFRAUCPY, set to one.

FANIN stage example



This example shows a complex pipeline where the **FANIN** stage collects all the output stages. The REXX program is as follows:

```
/****************************************************************************
 * Example of a complex pipeline using the FANIN stage
 */
PIPE (END +) NETV CDRMS
'| SEPARATE
'| LOCATE /'MSG'/
'| CHANGE /'MSG'//
'| PRESATTR RED
'| A: LOCATE /ACTIV/
'|+
'| PRESATTR GREEN
'| B: FANIN
'| CONSOLE
'+ A:
'++
'| C: LOCATE /NEVAC/
'| PRESATTR BLUE REV
'| B:
'+ C:
'++
'| B:
EXIT
```

All active resources (LOCATE /ACTIV/) are to be displayed in green. All never-active resources (LOCATE /NEVAC/) are to be displayed in blue. All other resources are to be displayed in red.

Student exercise

Student exercise



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 7.

Lesson 8: PIPE EDIT stage



Lesson 8: PIPE EDIT stage

- Modification of pipeline data
 - Create new message
 - Reformat
 - Modify line attributes (for example, color, line type)
 - Perform conversions (for example, SUBSTR, C2X)
 - Dynamically build commands
- PIPE EDIT capabilities
 - Global orders
 - Input orders
 - Output orders
 - Conversion orders

1-90

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

With the **EDIT** stage, you can make many types of changes, or edits, to a message within a pipeline. Edit data can include the following sources:

- Message data
- Line attributes
- Message attributes
- Command responses
- Literal data

With EDIT, you can create or reformat messages. In some cases, you can modify message and line attributes. You can edit as follows:

- Prevent calling REXX programs to manipulate messages.
- Improve performance. Editing within a pipeline flow is faster than driving a command to make the changes.
- Preserve message attributes when changing the message text.
- Improve programmer productivity when writing procedures to manipulate message data.
- Dynamically create commands to issue that are based on the contents of pipeline data.

EDIT is a simple stage consisting of global orders and edit phrases. Edit phrases define the action to be taken on the data as it flows in the pipeline.

- Global orders: Define the overall environment for the subsequent edit phrases. Global orders are optional.
- Input orders: Define the source of data to be processed by the conversion and output orders of the edit phrase.
- Output orders: Define how the resulting data is to be placed in the output line and, subsequently, on the output data stream.
- Conversion orders: Define how the data is to be manipulated. Conversion orders are optional.

PIPE EDIT stage example

IBM

PIPE EDIT stage example

```
Pipe netv WHO | DROP FIRST 1 | DROP LAST 1 | loc 1.9 /OPERATOR:/ |  
EDIT word 1 1 word 2 nw word 5 20 word 6 nw | color blue | coll | cons

LIST STATUS=OPS
OPERATOR: NETOP2 TERM: A01A701 STATUS: ACTIVE
OPERATOR: MVSAUTO TERM: MVSAUTO STATUS: ACTIVE
OPERATOR: AUTDVIPA TERM: AUTDVIPA STATUS: ACTIVE
OPERATOR: AUTOAON TERM: AUTOAON STATUS: ACTIVE
OPERATOR: AUTODC1 TERM: AUTODC1 STATUS: ACTIVE
OPERATOR: AUTODC2 TERM: AUTODC2 STATUS: ACTIVE
OPERATOR: AUTODC3 TERM: AUTODC3 STATUS: ACTIVE
...
STATION: NETOP2 TERM: A01A701
HCOPY: NOT ACTIVE PROFILE: DSIPROFB
STATUS: ACTIVE IDLE MINUTES: 0
...
END OF STATUS DISPLAY
```

1.91

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Explanation of PIPE EDIT stage in this example is as follows:

1. Issue the NetView WHO command. Delete the first and last lines in the response. Locate only the lines that begin with the string, OPERATOR:.
2. Use the PIPE EDIT stage:
 - a. WORD 1 1: Take the first word (OPERATOR:) from the input stream, and write it to the first word in *position* one of the output stream.
 - b. WORD 2 NW: Take the second word (for example, NETOP2) from the input stream, and write it to the *next word* of the output stream.
 - c. WORD 5 20: Take the fifth word (STATUS:) from the input stream, and write it to the output stream, beginning in *position* 20.
 - d. WORD 6 NW: Take the sixth word (for example, ACTIVE) from the input stream, and write it to the *next word* of the output stream.
3. Display the result in blue at the NetView operator screen. The result should be similar to the following text:

```
OPERATOR: NETOP2 STATUS: ACTIVE
OPERATOR: MVSAUTO STATUS: ACTIVE
OPERATOR: AUTDVIPA STATUS: ACTIVE
OPERATOR: AUTOAON STATUS: ACTIVE
OPERATOR: AUTODC1 STATUS: ACTIVE
```

OPERATOR: AUTODC2 STATUS: ACTIVE
OPERATOR: AUTODC3 STATUS: ACTIVE

And so on.

PIPE EXPOSE stage

IBM

PIPE EXPOSE stage

```
.-- RESPECT--.
|--EXPOSE--+-----+-----+-----+
    +- FORCE---+  '- NOLOG-'
    +- COMMAND--+
    '- TOTRAP--'
```

- Presents pipeline data to NetView automation:
- NetView exit DSIEX02A
- TRAP and WAIT processing
- Automation table
- NetView exit DSIEX16
- ASSIGN COPY routing
- Logging to the network log, system log, or hardcopy log, as appropriate

1-92 IBM Software Group | Tivoli Software
 © 2011 IBM Corp.

The **EXPOSE** stage causes messages in the pipeline to be exposed to automation by passing the pipeline messages to areas as follows:

- User exit DSIEX02A
- TRAP and WAIT processing
- Automation table for possible automation
- User exit DSIEX16
- ASSIGN COPY routing
- Logging (network log, system log, or hardcopy log)

You can specify actions as follows:

- COMMAND: Specifies that messages that the processing of a command generates in a previous CORRCMD, NETVIEW, or MVS stage are to be exposed. This action is to occur before the messages are absorbed into the pipeline.
- FORCE: Specifies that messages are to be exposed to user exit 02A, message automation, and user exit DSIE16. This action is to occur regardless of whether they have been previously exposed to those interfaces or not.
- NOLOG: Specifies that messages are to be processed as indicated by other specified keywords, but no logging is to occur.
- RESPECT: Specifies that messages that have been exposed to exit 02A, message automation, or exit 16 are not to be exposed to the same interfaces again.
- TOTRAP: Specifies that messages are to be exposed to only TRAP processing. With TOTRAP no logging occurs.

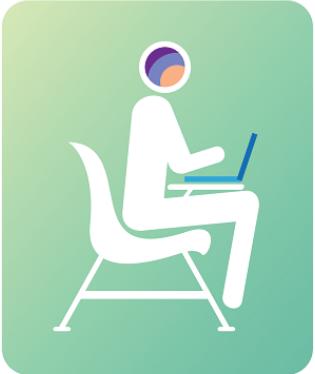
To expose the messages that the MYCLIST EXEC generates for trapping and waiting processing only, the text is as follows:

```
'PIPE NETV MYCLIST | EXPOSE TOTRAP | CONS ONLY'
```

Student exercise

IBM

Student exercise



1-93

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 8.

Lesson 9: Full-screen automation



Lesson 9: Full-screen automation

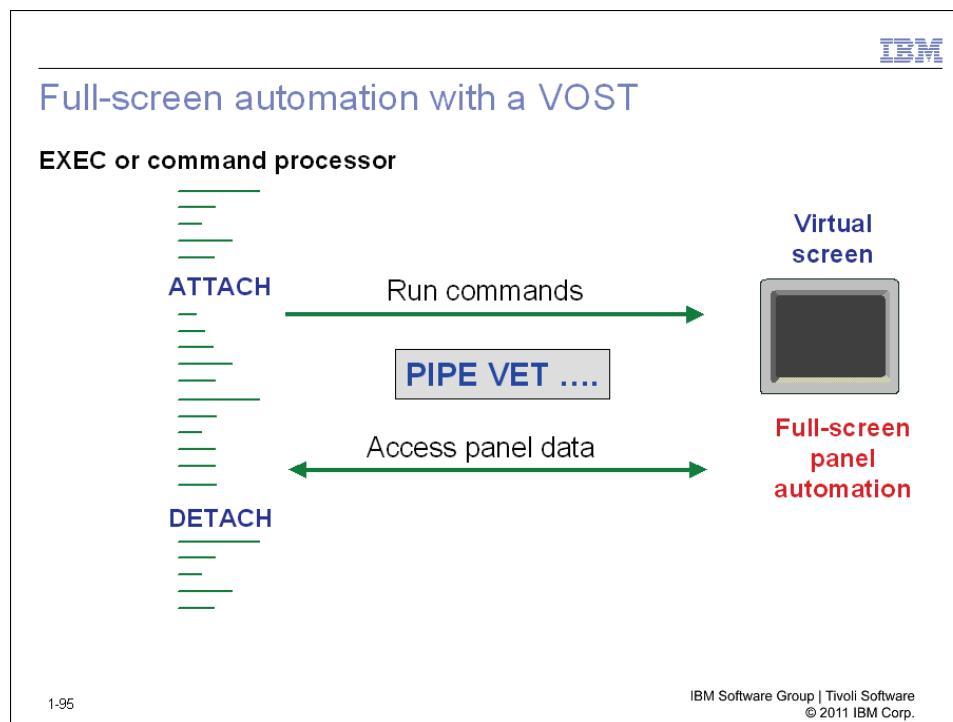
- Enables a program to interact with full-screen applications:
 - REXX, PL/I, or C
 - Simulation of an operator
- The program can perform as follows:
 - Reads data from an application panel
 - Writes data to an application panel
 - Simulates pressing keys on application panel
 - All PF keys, PA keys, Enter, or Clear

1-94

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Using a REXX, PL/I, or C program, you can connect to a full-screen application as a simulated operator. Under the simulated operator, you can issue commands and interrogate the screen for a response. This type of automation is typically called *screen scraping*. You can use a TAF full-screen session to connect to the application.

Full-screen automation with a VOST



This slide provides an overview of the basic steps to implement full-screen automation.

- The **ATTACH** command is used for beginning a simulated operator session, called a virtual OST (VOST), and to issue a command in that session. The virtual screen that is created on the VOST is 24 rows by 80 characters with no query support.
- The **PIPE VET** stage is used for reading data from and writing data to a virtual screen that is associated with a virtual OST (VOST). Row and field form data returns in a multiline message, BNH150I. NetView also provides a VET command.
- The **DETACH** command is used for ending the virtual OST (VOST) that an ATTACH command creates. DETACH simulates sending a LOGOFF command to the application that runs on the VOST.

See the *IBM Tivoli NetView for z/OS 6.1 Programming: Pipes* manual for more details. Online help is also available for ATTACH, DETACH, and PIPE VET commands.

NetView also provides examples of coding full-screen automation. Browse CNMS1101 in CNMSAMP or CNME2011 in CNMCLST to see sample REXX code.

Student exercise

Student exercise



IBM Software Group | Tivoli Software
© 2011 IBM Corp.

Open your *Student Exercises* book and perform Exercise 9.

Summary



Summary

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

- Discuss the basics of NetView PIPEs
- Use NetView PIPEs to perform the following tasks:
 - Issue commands
 - Trap and parse messages
 - Set and retrieve global variables
 - Read from and write to files
 - Perform automation

1-97

IBM Software Group | Tivoli Software
© 2011 IBM Corp.

IBM Tivoli certification and training

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

Certification

There are many reasons for certification:

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

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

Special offer for having taken this course

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

Role-based certification

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

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

Training

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

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



Printed in Ireland