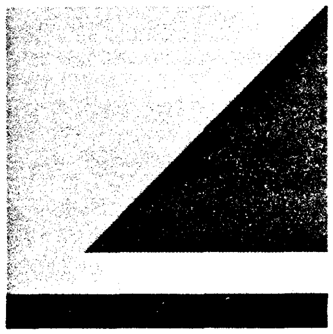
IBM®

NetView™

SC31-6016-2

Customization Guide

Release 3



File Number
S370/4300/30XX-50

Program Numbers
5685-152 (MVS/ESA)
5665-362 (MVS/XA)
5664-204 (VM)

Third Edition (August 1990)

This edition applies to Release 3 of the NetView licensed program, which runs under the following operating systems:

MVS/XA	(NetView program number 5665-362)
MVS/ESA	(NetView program number 5685-152)
VM	(NetView program number 5664-204)

Publications are not stocked at the address given below. If you want more IBM publications, ask your IBM representative or write to the IBM branch office serving your locality.

A form for your comments is provided at the back of this document. If the form has been removed, you may address comments to:

IBM Corporation
Department E15
P.O. Box 12195
Research Triangle Park, North Carolina 27709
U.S.A.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Special Notices

Programming Interfaces

This book is intended to help system programmers customize the NetView program. It contains descriptions of the parts of the NetView program that can be customized, and provides pointers to sources of background information. This book documents no programming interface for use by customers in writing programs that request or receive the services of NetView, except as noted below.

This book also documents general-use programming interface information and associated guidance information.

General-use programming interfaces allow the customer to write application programs that request or receive the services of NetView.

General-use programming interface information is explicitly identified where it occurs, either by an introductory statement to a chapter or section that is entirely general-use programming interface information, or by the following marking:

General-Use Programming Interface Information

Description of general-use programming interface information....

End of General-Use Programming Interface Information

Product-sensitive programming interfaces are provided to allow the customer installation to perform tasks such as tailoring, monitoring, modification, or diagnosis of this IBM product. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM product. Product-sensitive interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-sensitive programming interface information is explicitly identified where it occurs, either by an introductory statement to a chapter or section that is entirely product-sensitive programming interface information, or by the following marking:

Product-Sensitive Programming Interface Information

Description of product-sensitive programming interface information....

End of Product-Sensitive Programming Interface Information

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the Agreement for IBM Licensed Programs. Changes are made periodically to the information herein; before you use

IBM	MVS/ESA	MVS/XA	NetView
VM/XA	VTAM		

Contents

Figures

Tables

About This Book

This book is intended to help system programmers customize the NetView* program. It describes the parts of the NetView program that can be customized and provides pointers to sources of background information.

Use this book after you have installed the NetView program and discovered that a function or task needs to be created or modified for your use. You may need to create or modify panels, or write a user exit routine, command processor, command list, or subtask. This book contains procedures and examples to help you modify panels and develop your own programming enhancements.

To make full use of this guide, you need to understand how certain terms are used and where to find detailed information about those terms.

* Trademark of the IBM Corporation—see "Special Notices" on page iii for a list of IBM trademarks used in this book.

Terms Used in This Book

VTAM*
: ACF/VTAM, V3R2, V3R3

MVS
: MVS/XA*, MVS/ESA*

VM
: VM/SP, VM/SP HPO, VM/XA* (in compatibility mode).

Command List
: A list of commands and statements designed to perform a specific function for the user. Command lists can be written in REXX or in NetView command list language.

Command Procedure
: Either a command processor written in a high-level language (HLL) or a command list. See command processor and command list for further explanation.

Command Processor
: A user-written module designed to perform a specific function. Command processors, which can be written in assembler or a high-level language (HLL), are invoked as commands.

Component
: A command that (a) controls the terminal's screen (using the DSIPSS macro (TYPE=ASYPANEL) or the VIEW command), (b) allows the operator to enter NetView commands, and (c) can resume when such commands are complete.

High-Level Language (HLL)
: A programming language that does not reflect the structure of any particular computer or operating system. For NetView Release 3, the high-level languages are PL/I and C.

NetView Command List Language
: An interpretive language unique to the NetView program that is used to write command lists.

NAM
: Network Asset Management. An automated and tailorable means of collecting data from network resources for inventory and asset management.

REXX
: Restructured Extended Executor Language. An interpretive language used to write command lists.

User Exit Routine
: A user-written routine that receives control at predefined user exit points. User exit routines can be written in assembler or a high-level language (HLL).

VPD
: Vital product data. Product identification information such as machine type, model number, and serial number for hardware products. For software products, vital product data can be version and release level.

* Trademark of the IBM Corporation—see "Special Notices" on page iii for a list of IBM trademarks used in this book.

Customization Areas

This section discusses the areas of the NetView program that can be customized. Since customization of the NetView program takes place at various stages of network and system implementation, these topics are discussed in several different manuals. Table 1 on page xv indicates which NetView manuals contain more information on the listed topics.

Alias names are used to communicate across networks. You can use alias names to resolve duplicate resource names. With alias names, the name of the resource (such as a logical unit (LU), a class of service, a source LU (SRCLU), or a LOGON mode table) from the sending network is translated to a name that is unique to the receiving network. See *NetView Installation and Administration Guide* for more information on how to define alias names.

Filtering controls the amount of data presented to operators. It also controls the amount of data recorded. Message filtering using IF-THEN statements in the message automation table allows you to control the types of messages each of your network operators will receive. *NetView Installation and Administration Guide* discusses how to use automation statements to filter messages. IF-THEN statements are discussed in *NetView Administration Reference.*

Filtering can also be done on event data sent by network resources to the hardware monitor. *Recording filters* control which information is recorded into the hardware monitor's data base. Once data has been recorded, *viewing filters* can be set to determine which records will appear on each network operator's terminal. More information on hardware monitor filtering can be found in *NetView Operation.*

Generic alerts and code points are used to obtain problem determination support for devices and applications in your network that the NetView program does not automatically support. Chapter 4, "Customizing Hardware Monitor Displayed Data" on page 45 in this book contains details on how to use the IBM-supplied and user-defined code point tables to build hardware monitor Alerts-Dynamic, Alerts-Static, Alerts-History, Event Detail, and Most Recent Events screens.

You can use ***automated operations*** to implement automatic responses to events that occur in your network. *NetView Installation and Administration Guide* and *NetView Administration Reference* provide a more detailed explanation of how to define IF-THEN message automation statements to improve the productivity of your system/network operators. More detailed information on when and how to use the NetView program's automated operations can be found in *Console Automation Using NetView Planning* and *Console Automation Using NetView Implementing.*

National Language Support (***NLS***) will allow your operators to interact with the NetView program in a language other than English. The *NetView Installation and Administration Guide* explains how to write your own message translations in any language you prefer. The NetView program also offers a Kanji feature which supplies you with pre-translated panels and messages in Japanese using the Kanji character set. *NetView Installation and Administration Guide* discusses how to install the Kanji feature.

You may need to consider ***operator control and security***. To control who can gain access to the NetView program and what effect an operator can have on your network, you should consider some level of logon verification, scope of commands, and span of control. *NetView Installation and Administration Guide* discusses how to implement the different levels of security verification available in the NetView program. It also discusses how to limit which commands an operator can issue (scope of commands) and which part of the network's resources an operator can control (span of control).

You can create or change ***panels*** for your online help, online message help, help desk facility, the hardware monitor, and any user-written, full-screen applications. For a detailed explanation of how to create new panels or modify IBM-supplied panels for these components, refer to Part 2 of this book.

Sequential logging (Sequential Access Method Log support) allows you to write variable length records to multiple user-defined logs. These logs can be browsed or printed using your operating system facilities. Further explanation of how to define sequential log tasks can be found in *NetView Installation and Administration Guide, NetView Customization: Using Assembler* and *NetView Customization: Using PL/I and C.*

Session monitor data can be collected and kept in the session monitor data base. To control how much session data is collected and kept, you need to customize several session monitor definition statements. *NetView Installation and Administration Guide* explains how to change these definitions to fit the needs of your installation. Defining performance classes for the response time monitor (RTM) feature is also covered in *NetView Installation and Administration Guide.* Objectives and boundaries are set for each performance class and then a performance class is chosen for a session.

User-written functions add new functions to the NetView program or modify existing ones. You may want to develop your own command lists and user-written code. *NetView Customization: Writing Command Lists* discusses how to write command lists in REXX or in NetView command list language to help you control your network and make the operators' jobs easier. Information on how to write user-written code such as command processors and user exits in PL/I and C can be found in *NetView Customization: Using PL/I and C.* Information on writing command processors, user exit routines, and user subtasks in assembler language can be found in *NetView Customization: Using Assembler.*

Following is a list of the abbreviations and documentation names used in Table 1 on page xv.

Guide	*NetView Customization Guide*
Install	*NetView Installation and Administration Guide*
Op	*NetView Operation*
CL	*NetView Customization: Writing Command Lists*
PL/I, C	*NetView Customization: Using PL/I and C*
A	*NetView Customization: Using Assembler*
CA	*Console Automation Using NetView: Implementing*
	Console Automation Using NetView: Planning

Table 1. Customization Topics and Documentation

Topic	Guide	Install	Op	CL	PL/I, C	A	CA
Alias Names		√					
Automated Operations		√					√
Filtering:							
Message		√					
Hardware Monitor Filters			√				
Generic Alerts	√						
National Language Support		√					
Operator Control:							
Logon Security		√					
Scope of Commands		√					
Span of Control		√					
Panels:							
Hardware Monitor	√						
Help	√						
Help Desk	√						
User-written	√						
Sequential Logging	√	√			√	√	
Session Monitor Data:							
Response Time Monitor		√					
Session Awareness		√					
User-Written Functions:							
Command Lists (REXX, NetView)				√			
User-Written Programming (PL/I, C)					√		
User-Written Programming (Assembler)						√	

What Is New in This Book

Only minor changes have been made in this book. The changes include the addition of two small appendixes and a few other changes. The changes are identified by vertical lines in the left margins of pages where the changes appear.

Where to Find More Information

The following list shows all of the publications in the NetView Release 3 library, arranged according to related tasks. For more information on these and other related publications, see "Bibliography" on page 109.

Evaluation and Education	
Network Program Products General Information	GC30-3350
Bibliography and Master Index for NetView, NCP, and VTAM	GC31-6095
Learning about NetView: Operator Training (PC Diskettes)	SK2T-0292
Planning	
Planning and Reference for Netview, NCP, and VTAM	SC31-6811
NetView Storage Estimates (PC Diskettes)	SK2T-1988
Console Automation Using NetView: Planning	SC31-6058
Installation and Administration	
NetView Installation and Administration Guide	SC31-6018
NetView Administration Reference	SC31-6014
Network Program Products Samples	SC30-3352
NetView Tuning Guide	SC31-6079
Customization	
NetView Customization Guide	SC31-6016
NetView Customization: Writing Command Lists	SC31-6015
NetView Customization: Using PL/I and C	SC31-6037
NetView Customization: Using Assembler	SC31-6078
Console Automation Using NetView: Implementing	LY43-0007
Operation	
NetView Operation Primer	SC31-6020
NetView Operation	SC31-6019
NetView Command Summary	SX75-0026
Application Programming	
NetView Application Programming Guide: Program-to-Program Interface	SC31-6093
Diagnosis	
NetView Problem Determination and Diagnosis	LY43-0001
NetView Resource Alerts Reference	SC31-6024
NetView Problem Determination Supplement for Management Services Major Vectors 0001 and 0025	LD21-0023

Chapter 1. Designing and Implementing Your Own Functions

This chapter tells you what you need to know before making an addition or change to the NetView program, and lists some of the facilities available to help you customize tasks.

To customize NetView functions, you can add your own or modify the existing NetView program. Modifying existing functions include:

- Filtering or modifying the system management facility (SMF) records that the NetView program writes
- Providing your own policy to route operator messages
- Reformatting, analyzing, or compressing operator messages
- Checking command authority.

Additional function you may want to add could involve managing additional components in your network, such as X.25 data network components or voice network components. You can develop new applications and integrate them with the NetView program's existing management functions to meet your installation requirements. Examples of these user-defined functions might include:

- Real-time monitoring of specific resources, applications, or components in your network

- Collecting and recording additional SMF data for trend analysis or other data reduction applications you need

- Providing additional response time problem detection and alerting

- Detecting different classes of line problems and providing switched network backup (SNBU).

One approach to developing and implementing your own functions in the NetView program could be to:

1. Define your objectives by clearly expressing which changes or additional functions you need.

2. Understand how the new functions are to be used, and how they are to be provided.

3. Identify concepts and facilities in the NetView program that you can use to implement your functions.

4. Map your implementation into the NetView task structure.

5. Define transactions that implement your functions within tasks.

Defining Objectives

The first step in the design process is to understand the problem you are trying to solve. You may need to present new data to operators, change the way information is being presented, or provide another way to request information. To respond to such needs, you must decide which new operator commands are required, and the format you want to use to present new information. You may need to eliminate certain operator commands or actions thereby reducing the amount of information your operators must handle.

For some problems, you might focus more on data collection, data analysis, and data recording than on operator actions. Your objective may be to describe which network and data processing components are involved in input and output (I/O), and what processing is to be done on the data.

When you have clearly stated your problem and determined specific objectives for addressing it, you have in fact defined the solution as well.

Usage Scenarios

When you understand the problem and its solution, you should review the problem with the people who will be using your solution. Be sure the results you are trying to obtain match the needs of those whose problem you are trying to solve. You can adjust your solution, or you can gain a new understanding of the problem that may suggest a better solution. The more detail you can develop at this stage, the better. You should understand all the input needed, all the output to be produced, and all the steps required to collect, process, and record.

Identifying Conceptual Components

The functions required to support your usage scenario will be implemented within the components of the NetView program. You will need facilities to:

- Gather data on what is happening in your system and network
- Store and record some of this data
- Present data to NetView operators or to other programs.

Collecting Data

Typical sources available for collecting data that is useful in customization are:

- User exit interfaces provided in the NetView program.
- System or NetView services that provide status, configuration, processing, or authorization information.
- Data files and network devices that can be accessed using system or NetView services.
- Messages to operators that indicate important events are occurring in a system or an application.

User Exits

Some of the NetView program's exits allow access to network management data. Through these exits, user-written functions can obtain the text of operator commands, messages, and logons. Data that the NetView program writes to VSAM files and to the SMF log, as well as data on the VTAM CNMI, can be accessed within other NetView exits. Refer to *NetView Customization: Using Assembler* and *NetView Customization: Using PL/I and C* for more information on these user exits.

Service Routines

System or NetView services give you access to information such as the system date and time, the addresses of programs, the addresses of named storage areas, operator span of control and scope of authority, and the values of command list variables. Refer to *NetView Customization: Using Assembler* for information on DSIDATIM, DSICES, DSIFIND, DSISSS, and DSIKVS. Refer to *NetView Customization: Using PL/I and C* for information on CNMINFC, CNMNAMS, CNMSCOP, and CNMVARS.

Data Files

The NetView program provides specialized disk services and VSAM data services to access network management data files. In addition to these, functions written in high-level language (HLL) may invoke system allocation and access methods to read and write data. CNMI services also provide access to data coming from devices in the network. Refer to *NetView Customization: Using Assembler* for information on DSIDKS, DSIZVSMS, DSIZCSMS. Refer to *NetView Customization: Using PL/I and C* for information on HLL VSAM and CNMI services. Refer to *NetView Customization: Writing Command Lists* for information on REXX file I/O.

Operator Commands and Messages

Operator commands can be issued within command procedures to request status data. The resulting response messages containing the requested status data can then be trapped and processed in the command procedure. Data in other system and network messages can also be processed in user-written command procedures that can be invoked through NetView message automation. Refer to *NetView Customization: Writing Command Lists* for information on REXX and NetView command list language message processing. Refer to *NetView Customization: Using PL/I and C* for information on HLL message processing.

Data Storage and Recording

NetView command list variables can be used to store and retrieve data needed for many user-written functions. Command procedures written in REXX, NetView command list language, PL/I, or C may create, set, and read command list variables.

For permanent storage and for larger volumes of data, you may want to record certain information in data files rather than simply naming it and storing it as a command list variable. The NetView program provides logging facilities for recording this data in a log. For example, you may find it useful to log activities of your own applications along with system or network activities that the NetView program is logging. In other cases, you may want to produce a separate log of your own for data that you collect. Refer to *NetView Installation and Administration Guide* and "Choosing an Application Program Interface" on page 9 in this book for information on sequential logging.

Operator Presentation

Some of the NetView program's operator presentation functions can be customized or extended by using the VIEW command or by modifying panels that are used by some components of the NetView program to present data to operators. Refer to Chapter 2, "Using the VIEW Command" on page 13 and Chapter 3, "Modifying and Creating Online Panel Interfaces" on page 35 for more information on these topics.

You can also use messages as a means to present information to operators. By presenting data from user-written functions in the form of messages, the data becomes subject to message automation processing, allowing both automatic and manual operation of your functions. Refer to *NetView Customization: Using Assembler* for DSIWCS, DSIMBS, DSIMQS, DSIPSS, and other message services. Refer to *NetView Customization: Using PL/I and C* for CNMSMSG. Refer to *NetView Customization: Writing Command Lists* for REXX and NetView command list language write-to-operator (WTO) messages and other message services.

Task Structure

In order to write your own functional extensions to the NetView program, it is necessary to first understand how the NetView program operates. The NetView program's design is based on MVS system concepts and facilities. In a VM system environment, the guest operating system group control system (GCS) is used to simulate MVS system services for both VTAM and the NetView program. *MVS/XA Supervisor Services and Macros* is a good reference for further explanation of how words such as *dispatch*, *task*, and the names of various system services are used in this discussion.

The NetView Program as a System Application Program

First consider how the NetView program looks to the operating systems that support it. The NetView program is organized into several tasks running in parallel, each one separately dispatchable in a multitasking environment. When any one task is idle, any of the others are eligible to run. A system multitasking dispatcher is used to accomplish this by using the NetView program's ATTACH system service to create each new task. The WAIT system service is invoked by each NetView task when it has no more processing work to do and is ready to become idle. The POST system service is used to take any task out of an idle state and make it dispatchable when that task has new input data ready to be processed.

The NetView Program Tasks

When the NetView program starts, its main task attaches several subtasks of different types depending on the function that each is to perform and on the name of the program that each task runs when it starts. Each different task type determines the specific system interfaces and operator interfaces that are available under that task, and the type of transactions that may be performed under the task.

Each operator station task (OST) supports one NetView operator identified by a unique name. The operator identifiers (OPIDs) are defined in the NetView parameter library and are assigned to an OST when an automated operator, known as an autotask, is activated using the AUTOTASK command, or when an operator logs on using a VTAM connected terminal. The program module DSIOST runs for every OST.

Each NetView–NetView task (NNT) also supports an operator. This type of task is used when the operator logs on to the NetView program from another NetView program instead of from a terminal. The other NetView program may be running in a different machine but must be connected through VTAM. The operator logs on from the other NetView program using the START DOMAIN command. The program module DSIOST also runs for every NNT.

Each hardcopy task (HCT) supports a 3287 printer connected through VTAM to provide a hardcopy log for one or more operators.

There is only one primary programmed operator interface (POI) task (PPT) for each NetView program, and it runs the program module DSIPPT. The PPT provides time interval notification for all NetView tasks. When VTAM is running, the PPT opens a special VTAM application control block (ACB) for the VTAM programmable operator interface (POI) to receive unsolicited data from VTAM.

Each optional task (OPT) must be defined by a TASK statement in the NetView parameter library. The program module that runs for an OPT may be any program that meets the specification for optional tasks described in "Adding Optional Tasks to the NetView Program" on page 8.

Each data services task (DST) is a specific case of an optional task whose program module is DSIZDST. (See "Adding Optional Tasks to the NetView Program" on page 8.) The TASK statement for a DST may name an initialization member in the NetView parameter library from which statements are read to define parameters for the functions performed by the specified DST.

Program Activity within a Task

All the NetView tasks discussed in "The NetView Program Tasks" on page 4 share a common program design. They are all designed to WAIT once they have been started until they receive a POST that causes them to be dispatched to perform a specific unit of work. When that work is complete, the WAIT macro is issued again and the task continues running only if and when another POST has been done for another unit of work. All these tasks use a list of event control blocks (ECB) when they issue their WAIT. The NetView customization macros and services are provided to ensure that any implied waiting is done through the task's ECB list so that all of the NetView program's task-request interfaces remain enabled.

Every NetView task has its own termination ECB and its own message queue ECB. Depending on the type of task, (OST, DST, etc.) there may be additional ECBs in a task's ECB list that represent different kinds of processing that the task has been designed to test for and perform when it is POSTed out of its WAIT state.

Queuing Work to NetView Program Tasks

While any one task is in its normal WAIT state, any of the other tasks in the NetView program may be running. In fact, even a running NetView task may at any time be interrupted due to an event in the system, and may be preempted by a higher priority task until that task issues its normal WAIT. System functions outside of the NetView program may also interrupt the NetView program processing by scheduling interrupt exit routines to run that are associated with specific NetView tasks.

As a result, a task may have data placed in its message queue or another work queue that it is designed to support, and it may be POSTed to perform that work at any time. The data may have originated in another NetView task. This commonly happens when a DST queues message data to an OST to be displayed to an operator. The data may have come into the NetView program through an interrupt exit routine which was scheduled by an event such as a VTAM RECEIVE request being completed from a NetView operator's terminal. This is how the text of an operator command is given to an OST to be processed.

Message and Command Buffers

The data placed in the various task queues is formatted into a special data structure commonly referred to as a *message buffer* or *buffer*. A buffer header at the beginning of the buffer indicates the type of data that it contains and any special formats by which the data must be accessed. Commands are processed by programs called command processors which you provide as a part of your customization programming for the NetView program. Messages are processed either according to predefined processing built into the NetView task, or by message automation command processors. Where a message or other data in the NetView program is not handled in the command processor interface, your customization programming may provide user exit routines to determine some of the processing that is performed on this data.

Immediate Commands

An immediate command starts processing as soon as an operator enters the command, regardless of any regular command currently running. Thus the requested function is performed at once, even if the task is in the middle of a large queue of work.

An immediate command runs under the OST (operator station task) and NNT (NetView – NetView) subtask environments. Unlike other commands, immediate commands receive control with the TVBINXIT bit set on. This means that they interrupt mainline processing and cannot be interrupted by another command but may be interrupted by other exits in asynchronous activity.

Long Running Commands

A long running command is a command that is able to suspend processing to allow other activity, such as operator commands and data retrieval, and then resume. All the NetView components are long running commands. NetView command list language, REXX and HLL command procedures are long running commands. For an assembler command, the DSIPUSH macro provides the capability to run as a long running command.

Long running commands run under an OST, NNT, PPT, and (limited) DST. They may be invoked directly by operator input, called by a command list, or called by another long running command. They return control to the NetView program after scheduling work but before processing is complete. The NetView program then processes other work that may be pending.

Long running command processors are often used to retrieve data from another task or from another domain without allowing the calling function or calling command list to proceed in the midst of this retrieval. During this retrieval, the processor's task may continue to receive messages and accept commands.

Data Services Commands

A data services command processor (DSCP) runs under the DST subtask environment. DSCPs perform CNM data services, or VSAM data services, or both. DSCPs are also appropriate for centralized or serialized user-defined functions that do not use CNMI or VSAM services.

Defining Transactions — Exits and Commands

There are two types of transaction programs you may provide within the NetView task environments: user exits and command processors. The programming interface details are provided in *NetView Customization: Using PL/I and C* and *NetView Customization: Using Assembler*. In designing user-written functions you will use the user exit interface and the command processor interface in the NetView program to fit your own programming into the overall structure of the NetView program.

User Exit Programs

Global user exits are provided in the NetView program at several points in the processing of logon and logoff data, command buffers, and message buffers. Different exits are driven based on the origin of the buffer and the stage of the NetView program processing that the exit is in. Special exits are driven under DSTs to handle data during initialization, input, and output under these tasks. For a summary of the NetView user exits refer to *NetView Customization: Using Assembler* and to *NetView Customization: Using PL/I and C*.

General user exits are identified and invoked with preassigned module names of DSIEX*nn*, and the DST exits are identified uniquely for each DST in its task DSTINIT initialization statements.

Command Processors

NetView command processors can be started as a result of:

- An operator request
- A command buffer queued to a task for processing by any program in NetView
- A command call from another command processor
- An action specified in the message automation table.

A command procedure may be written in NetView command list language, REXX, PL/I, C, or assembler language. Those that are written in NetView command list language or REXX are defined as command processors to the NetView program by placing them into the NetView command list library (ddname DSICLD). Other command processors must be link-edited into the NetView load library (ddname STEPLIB), and must be defined to the NetView program with a CMDMDL statement in the DSICMD member of the NetView parameter library (ddname DSIPARM).

You may implement a function as a number of different user exit programs and command processors, each designed to perform a separate part of the overall function. A common split of function across command processors is to divide processing between OSTs and DSTs. Since OSTs receive data from operator stations and present data back to them, a command processor is written:

- to be called when the command is entered by an operator
- to parse the command data and form a data services request
- to queue a command buffer containing the data services command to be processed by the DST
- to provide an error message or a command confirmation message back to the operator.

The DST completes the function in a separate command processor that is called as a result of the command buffer that was built and queued by the first command processor. Under the DST, functions requiring the special data services of VSAM, external logging, or the VTAM CNMI are performed and messages may be issued back to whichever operator task queued the command. Figure 1 on page 8 shows a typical program design for a function which uses the CNMI and VSAM services.

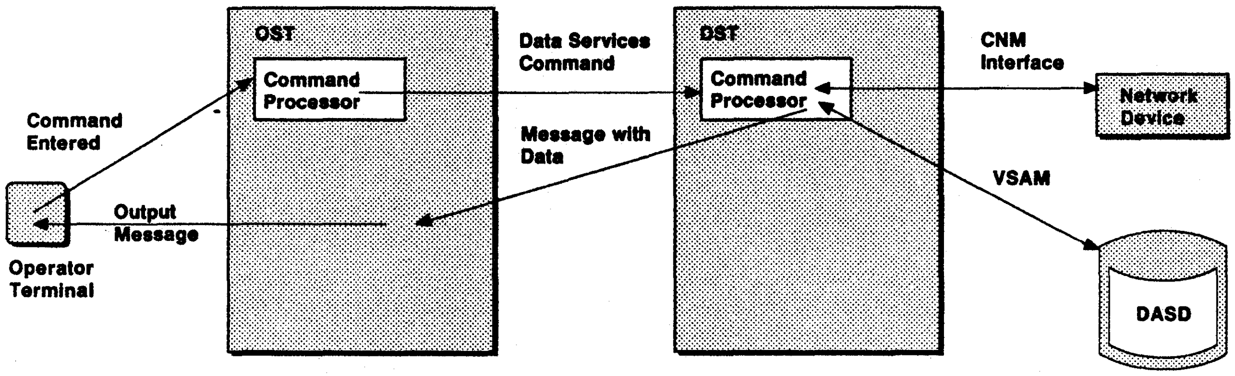


Figure 1. Program Design Example for DST Function

Another important method for breaking up a complex function into a sequence of separate transactions is called Long Running Command support in the NetView program. Command processors may use this support to establish a named stack entry where an anchor address can be saved and retrieved. In this way, one command processor may perform some processing and save its results in storage. A related command processor may later retrieve the address of this storage from the long running command stack and perform another phase of the same processing.

Adding Optional Tasks to the NetView Program

The recommended method of writing a user subtask is to use the existing DST subtask as a base. This is the only method supported in PL/I and C. The other, more complex method is to write a completely new subtask in assembler language, which the NetView program starts as an optionalOPT, or subtask. For an OPT, you must supply code for the subtask's initialization, exit, message, and command processing functions and termination. Since some of these elements are already provided in an existing DST subtask, using the DST as a starting point is generally the more practical approach. For more information on OPTs and DSTs in assembler see *NetView Customization: Using Assembler.* For implementing DSTs in high-level language, see *NetView Customization: Using PL/I and C.*

Choosing an Application Program Interface

One application program interface may be more suitable than another for your particular customization requirement. The effects on performance, ease of creation, and maintenance should be considered when determining which interface to use.

Performance

Performance-critical applications should be written in a compiled language. Generally, compiled or assembled command procedures execute faster than interpretive (REXX and NetView) command lists.

NetView-driven exit routines must be written in a compiled or assembled language. Any command processor that accesses NetView control blocks must be written in assembler. Command procedures that can be driven by terminal input or by messages and do not need to access NetView control blocks can usually be written in REXX or in NetView command list language.

Preloading a REXX or NetView command list (via the LOADCL command, see *NetView Operation* for information on this command) improves overall performance of the command list when compared to a command list which has not been preloaded. For additional performance recommendations, see *NetView Tuning Guide* and *NetView Installation and Administration Guide.*

Ease of Creation

REXX and NetView command lists are usually easier to create and debug than compiled command procedures. This is primarily due to the interactive debug capability, the availability of higher-level functions and commands, and the fact that you need not consider variable data typing.

Maintenance

Updating and maintaining REXX and NetView command lists is easy because the changes can be incorporated and tested online without having to restart the NetView program. Compiled command procedures must be link-edited into a load library and may require the NetView program to be restarted depending on whether the module is loaded for each execution. Table 2 on page 10 lists additional application program interface capabilities.

Table 2. Selecting an Application Programming Interface

Function	**Command List**	**PL/I or C**	**Assembler**
Send message to NetView operator in line mode	Yes	Yes	Yes
Interact with operator through NetView operator screen (PAUSE/GO command)	Yes	Yes	No
Invoke NetView commands	Yes	Yes	Difficult
Trap and process messages destined for an operator	Yes	Yes	Difficult
Access task and common global variables	Yes	Yes	No
Create and access named areas of storage	No	Yes	Yes
Interact with operator through full screen panels	via VIEW	via VIEW	Difficult
Communicate non-SPCI data over the CNMI	No	Yes	Yes
Access DASD files	Limited	Yes	Yes
Program debugging support provided	Yes	Yes	No
Implement NetView user exits	No	Yes	Yes
Access NetView control blocks	No	No	Yes

Logging

The NetView program provides several methods for logging information. Table 3 indicates features of the common logging methods available.

Table 3. Features of NetView Logging Methods

Feature	**Network Log**	**External SMF Log (MVS only)**	**External User-Defined Log**	**NetView Sequential Log**
Access method	VSAM	VSAM	Sequential	BSAM
Device-independent	No	No	Yes	Yes
Function provided	Record all operator station activity	Service level verification and accounting	User-defined	Base service for user-defined functions
API – HLL	CNMSMSG	CNMSMSG	CNMSMSG	CNMSMSG
API – Assembler	DSIWLS	DSIWLS	DSIWLS	DSIWLS
Begin recording	START	See *NetView Installation and Administration Guide*	See *NetView Installation and Administration Guide*	See *NetView Installation and Administration Guide*
Browse	NetView BROWSE	No	Operating system browse	Operating system browse
Multiple log tasks	No	No	No	Yes
Variable length blocks and records	No	Yes	Yes	Yes
Primary/secondary data sets or files	Yes	System controlled	No	Yes
SWITCH, RESUME, AUTOFLIP	Yes	N/A	No	Yes (No RESUME in VM)
User exits	Many	XITXL	XITXL	XITBN, XITBO

For more information on the Network log, see *NetView Problem Determination and Diagnosis* and *NetView Installation and Administration Guide*. Information on external logging using either System Management Facility (SMF) or a user- defined log or sequential logging refer to *NetView Installation* and *Administration Guide*.

Chapter 2. Using the VIEW Command

This chapter describes the VIEW command processor, which is used to display full-screen panels from user-written programs. This chapter provides general-use programming interface information. (General-use programming interfaces are defined in "Special Notices" on page iii.)

Using VIEW, users can design their own panels controlling the color and highlighting of panel text.

The VIEW command allows command lists or command processors written in a high-level language (HLL) to interact with an operator by means of full-screen panels. The data from the command list or HLL variables can be substituted into the panels.

You can use VIEW to create a rollable application. See "Creating a Rollable Component with VIEW" on page 19. The panel being displayed can contain multiple input fields whose contents can be returned, along with the Attention Identification (AID) information, to the calling command list or HLL command procedure. Incoming messages can dynamically update the contents of the panel being displayed.

The NetView program supplies a number of command lists that use the VIEW command to display full-screen panels. Displaying a new panel by invoking VIEW from a command list requires that you either modify an existing command list or write a new one. (If you change an IBM-supplied command list, you should first copy it into your own command list data set and change its name.)

When modifying or writing a command list for your new application or sequence of panels, use the VIEW command and specify NOINPUT to display online help panels. See Chapter 3, "Modifying and Creating Online Panel Interfaces" on page 35 for more information on how to code help panel hierarchies. The VIEW command is intended to be used only from a command procedure. In addition, the SHOWCODE command list may be used to display VIEW's nonzero return codes and associated explanatory messages. See "Displaying VIEW Return Codes" on page 33 for details. Code the command as follows:

VIEW *name1 name2* [{**MSG**|**<u>NOMSG</u>**}][{**INPUT**|**<u>NOINPUT</u>**}]

name1
A name (1 to 8 characters) to be used internally by the NetView program. The first character must be alphabetic. A distinct name should be used for each separately rollable application. For compatibility with prior releases of the NetView program, characters 3-9 are allowed.

name2
The name (1 to 8 characters) of the panel to be displayed.

MSG
If MSG is specified, the panel display is interrupted by *any* message received by the operator station task (OST) under which the VIEW command is running. This allows the contents of the panel being displayed to be refreshed without requiring operator intervention. An incoming message can drive a command procedure that updates some of the variables substituted into the displayed

panel. When the command procedure completes, the VIEW command processor resumes, and the contents of the displayed panel reflect any changes to the variables made by the command procedure. If MSG is specified, the panel can be refreshed while entering data.

NOMSG
When NOMSG is specified, incoming messages do not interrupt the panel being displayed. This is the default.

INPUT
Returns input values, AID information, and cursor location to the procedure invoking the VIEW command. When using the VIEW command with the INPUT option, always issue the UNIQUE command first. If the INPUT option is not specified, using the UNIQUE command is optional. See "Using the UNIQUE Command" on page 20 for more information.

NOINPUT
When the NOINPUT option is specified, the VIEW command does not return any information to the procedure that invoked it. This is the default. If the panel defines a command line, input is treated by the NetView program as a command. See Figure 10 on page 36 for the PF keys provided by the NetView program when NOINPUT is specified.

Panel Definition for Use with VIEW

The definition statements described in this section are those that are common to the VIEW command with either INPUT or NOINPUT options specified.

Controlling Color and Highlighting of Fields

You can change the color and highlighting of existing panels or add color and highlighting to panels you create. Text color and highlighting in the displayed panel are controlled by attribute symbols or attribute variables. Once attribute symbols are coded in the source panel, they appear as blanks in the displayed panel.

Scanning for attribute symbols or variables in a particular line occurs only if column 1 contains an attribute symbol, or column 1 contains a variable which has an associated attribute variable. Otherwise, the line is displayed *as is* from the panel description in the default color and without variable substitution.

Note: Color and highlighting depend on the kind of terminal you are using.

Attribute Symbols

Attribute symbols are defined on the source panel to color or highlight text. Edit the source panel and replace the blank space before the text with the appropriate attribute symbol selected from the second column of Table 4 on page 15, or Table 5 on page 15.

An option specified in the header of each panel determines which set of attribute definitions should be used for that panel. If no option ("****") is specified, the original set (Attribute Set 1) of special character interpretations is used. Attribute Set 2 is used when specified on the text indicator line of the panel definition ("*** AT2").

Table 4. Set 1 Color and Highlighting Attributes

Attribute Set 1	Symbol	Hex Character	Intensity	Field
White	%	X'6C'	High	Text
Reversed White	}	X'D0'	High	Text
Underscored White	!	X'5A'	High	Text
White	~	X'A1'	High	Input
Turquoise	$	X'5B'	Normal	Text
Underscored Turquoise	\	X'E0'	High	Text
Blue	+	X'4E'	Normal	Text
Reversed Blue	{	X'C0'	High	Text
Green	@	X'7C'	Normal	Text
Yellow	¬	X'5F'	Normal	Text
Pink	¦	X'6A'	Normal	Text
Red	¢	X'4A'	High	Text

Table 5. Set 2 Color and Highlighting Attributes

Attribute Set 2	Symbol	Hex Character	Intensity	Field
White	%	X'6C'	High	Text
Reversed White	}	X'D0'	High	Text
Reversed Red	!	X'5A'	High	Text
White	~	X'A1'	High	Input
Turquoise	$	X'5B'	Normal	Text
Reversed Green	\	X'E0'	Normal	Text
Blue	+	X'4E'	Normal	Text
Reversed Blue	{	X'C0'	Normal	Text
Green	@	X'7C'	Normal	Text
Yellow	¬	X'5F'	High	Text
Reversed Yellow	¦	X'6A'	High	Text
Blinking Red	¢	X'4A'	Normal	Text

Displaying Special Attributes

If you want a particular symbol that doubles as an attribute to be displayed within a colored or highlighted row, place double quotation marks (") in front of the symbol. For example, if you want the left brace ({) to appear in text, type "{ in the source panel. If you want double quotation marks (") to be displayed, precede them with another set of double quotation marks; thus, type "". When the same hexadecimal values for these symbols are coded as part of Kanji text surrounded with shift-out and shift-in, they will not be treated as attributes.

Be careful how you use the attribute for the color blue, the plus sign (+). If you want to assign the color blue to a variable defined by the NetView command list language, you must enclose the plus sign in single quotes (&COLOR = '+'). Otherwise, the NetView program interprets the plus sign as a continuation character.

A special rule also exists for use of the dollar sign ($) and the "at" sign (@) within variable values. When a variable has a string value containing the dollar sign or the "at" sign, those characters will not be interpreted as color attribute symbols; they will instead be displayed. To obtain the color attribute effects of these symbols within variables, use the less than (<) and the greater than (>) symbols respectively (&COLOR= <$; &COLOR= >@). When not within variable values, the less than and the greater than symbols are displayed.

Attribute Variables

Attribute variables are assigned in the command procedure which drives the view panel. An alternative to defining attribute symbols on the panel or within the variable data is to define attribute variables that are associated with panel variables. Attribute variables describe attributes associated with panel variables and their following text on the same line. Using an attribute variable provides a wider range for attribute selection and allows definition of input fields. When using an attribute variable, the contents of the associated panel variable are not scanned for attribute symbols.

An attribute variable name is formed by concatenating a dollar sign onto the *front* of the panel variable name. For example, in NetView command list language, the attribute for panel variable &V1 is defined in a variable called &$V1. In REXX and HLL, the & is not used.

The following is the syntax for the contents of an attribute variable:

```
attribute variable ='tv tv tv...'
```

where tv is the *type value* pair. Multiple pairs of the same type in one attribute variable are allowed; the last one is accepted and the previous ones are ignored. The values for *type value* are as follows:

tv = type value

- **F** = Field
  - **FA** Protected - data cannot be entered on displayed panel, this is the default
  - **FI** Unprotected - data can be entered on displayed panel
- **I** = Intensity
  - **IN** Normal intensity; default if no intensity value is specified
  - **IH** High intensity
  - **ID** Dark, nondisplayable
- **C** = Color
  - **CD** Default color for device; default if no color value is specified
  - **CW** White or neutral
  - **CB** Blue
  - **CR** Red
  - **CP** Pink
  - **CG** Green
  - **CT** Turquoise
  - **CY** Yellow
- **H** = Highlight
  - **HD** Default extended highlighting; also, default if no highlighting value is specified
  - **HB** Blinking
  - **HR** Reverse video
  - **HU** Underscored
- **U** = Cursor
  - **UN** Cursor will not be placed at this field, this is the default.

UY Cursor will be placed at the beginning of this field. UY specifications for multiple variables cause the last one specified to be accepted and the previous ones to be ignored.

Note: If no cursor is specified for any attribute variable on a panel the cursor will be placed at the beginning of the first input field.

Consecutive blanks between *type value* pairs define a delimiter. The following is a NetView command list language example where &V1 is defined as a protected field with high intensity in red. &V2 is defined as a protected field in high intensity, in turquoise, with the cursor placed here.

```
&$V1 = 'FA IH    CR'
&$V2 = 'IN IH CT UY IH'
```

In the following REXX example, V1 is defined as an input variable (unprotected field) with no cursor. For V2, all the defaults are used.

```
$V1 = 'FI  UN'
$V2 = ' '
```

Attributes defined by attribute variables or attribute symbols apply until one of the following occurs:

1. The end of the line, or
2. The explicit placement of an attribute symbol later in the line, or
3. A variable is encountered later in the line that has either:
   a. A valid attribute variable that specifies new attributes, or
   b. No valid attribute variable but contains one or more attribute symbols.

Constants or variables defined on a panel can become part of an input field and are updated only when you type over some portion of the input field. When you type on an input field, the entire contents of the input field are assigned to the panel variable.

The first byte of a field defined by a panel variable (the &) is used for attribute specification, and is followed by the contents of the variable. If an attribute variable corresponds to a panel variable, it takes effect at this first byte even if the panel variable is not found (and is replaced by blanks).

Trailing blanks in a field defined by the panel or caused by variable padding are replaced by nulls on the output panel. A distinction is made between trailing blanks in the variable (not replaced) and those caused by panel definition or padding (replaced by nulls). For example, if &VAR contains the value z and the panel line containing &VAR included &VAR !, the blank following the Z is output as a blank while the rest of the field up to the attribute symbol (!) is output as nulls. This allows you to use the insert key for character insertion in an input field.

Note: If an attribute variable contains a syntax error, message CNM944I is written to the NetView log if it is active.

Displaying Variables in Source Panels

When VIEW attempts to resolve a variable name coded on the panel definition statement, it checks first to see if the variable is a NetView control variable. If the variable name is found, the appropriate NetView control variable is substituted. Otherwise, VIEW searches for a variable of the same name defined by the command procedure that invoked the VIEW command. For example, if the variable name

&OPID is coded in the panel definition, the value for the control variable &OPID is always substituted instead of the value of a command procedure variable named OPID. If a variable name is not defined to the NetView program or defined in the calling command procedure, the variable coded on the panel is displayed as a string of blanks.

You can use only 1 to 11 alphanumeric characters (A - Z and 0 - 9) for the variable names in VIEW panel definitions. Alphabetic characters must be in uppercase. Variable names also must conform to any other variable naming conventions set by the language invoking VIEW if the variable is to be referred to by that language. For example, variable names used in HLL and REXX must start with an alphabetic character.

For the VIEW command to find global variables, the global variables must have been referenced by the command procedure prior to executing the VIEW command. Global variables are defined by &TGLOBAL or &CGLOBAL in NetView command list language, GLOBALV in REXX, CNMVARS in PL/I, or *Cnmvars* in C. See *NetView Customization: Writing Command Lists* for more information on global variables.

Note: A REXX user who calls VIEW to display global variables, or to allow overwriting global variables by defining them in an input field, must:

1. Issue a 'GLOBALV GETT (or GETC) *varname*' before invoking VIEW.
2. Initialize *varname* or make sure it has a non-null value before invoking VIEW.

If the previous steps 1 and 2 are done, the global *varname* is displayed and is updated if *varname* has an attribute variable that makes it an input field. Otherwise, the REXX local *varname* is displayed and updated. When VIEW accesses a global variable this way, any REXX local variable with the same name is not affected by VIEW.

If a NetView control variable (for example, APPLID or OPID) is named on a VIEW panel, VIEW displays the control variable value and cannot access a REXX local variable with the same name. Control variables cannot be updated.

The following REXX example shows how VIEW can be used to allow the operator to update a global variable.

```
/*   */
'GLOBALV GETT XYZ'
IF XYZ = '' THEN
 DO
   XYZ = '    '
   'GLOBALV PUTT XYZ'
 END
$XYZ = 'FI'
VIEW NAME1 TESTPANL INPUT
'GLOBALV GETT XYZ'
SAY XYZ
EXIT
```

If the length of the value assigned to the variable exceeds the length of the variable in the source panel, and if the variable is followed by alphameric or special characters (such as !, ¢, \, ¦, @, #, $, %, ¬, &, ", +) on the panel definition, the value is truncated. When a variable is followed by characters other than these (such as a period or a dash), the characters are overwritten.

You can assign a particular color to a variable by defining an attribute variable. See "Full-Screen Input Capabilities" on page 21 for an example showing how variables in source panels are displayed.

Creating a Rollable Component with VIEW

A NetView component is a command or command procedure that controls the terminal's screen, provides for operator entry of arbitrary NetView commands, and is capable of resuming when such commands are complete. In a command procedure, you can create a rollable component using VIEW to provide the necessary screen control.

If you specify the NOINPUT option, and your panel has a command line (see page 37), VIEW handles the operator command interface for you. If you specify the INPUT option on your VIEW command, you provide a command line at the bottom of your panel. VIEW returns the operator's input to your procedure in the form of named variables.

Issuing Commands from Command Procedures

When a command is issued directly from a command procedure, the procedure is suspended until that command completes. When the called command is complete and the return code is available, the procedure resumes execution. If the called command is a long running command, it and the calling procedure form a group that is treated as a unit by the NetView program's ROLL command (roll group).

Note: An exception is the BGNSESS FLSCN command, which allows a calling procedure to complete before the session begins. This is done by using the MINOR option of DSIPUSH. (See *NetView Customization: Using Assembler* for information on DSIPUSH.)

Such grouping is beneficial if the intent is to build a hierarchy of related panels, using different procedures to build each one. The grouping is not desirable when executing unrelated commands, such as those received from an operator.

To disassociate an unrelated command from the calling procedure, use the CMD command. For example, if an operator's command is the value of a variable *cmdline* in your REXX procedure, issue `'CMD HIGH'` *cmdline* to queue the command and to return control to your command procedure. When the queued command is processed, it is asynchronous and not related to current processing. Thus, if an operator types STATMON on your panel's command line and then issues ROLL from STATMON, your procedure regains control even though STATMON is not complete. Queuing, rather than calling a command, protects your procedure from any reset condition this command might encounter. (See *NetView Customization: Using Assembler* for more information on the ROLL function.)

Using the UPPER Command

When issuing a command that was specified from a VIEW panel when the INPUT option was specified, and the application is coded in a high-level language (HLL) or in the NetView command list language, the command MUST BE translated into uppercase in order for the NetView program to recognize it. This can be accomplished with the UPPER command. The UPPER command changes the contents of the specified variables to uppercase. An example of the UPPER command can be found in Figure 4 on page 23. Code the command as follows:

```
UPPER  variable var1,...var2
```

variable var1,...var2 the 1- to 31- character name of the variable to be translated to uppercase.

Usage Notes:

When coding in the NetView command list language, do not specify the leading "&" in front of the variable name.

When more than one variable is specified, all variables will be translated, even if one of the variables has an error condition (not found or invalid length).

Return Codes:

0	Successful completion of all specified variables.
4	At least one variable not found.
8	At least one variable length not within range.
12	At least one variable not found and at least one other variable length not within range.
16	Not invoked from a command procedure.
20	No variables specified.

Using the UNIQUE Command

The NetView program allows an operator to start many copies of the same command processor. In some cases, you may not want more than one copy, such as when creating a NetView component. By using DSIPOP or DSIPUSH with the PROMOTE option, assembler programmers guarantee the uniqueness of their long running commands. Using the UNIQUE command will achieve the same type of uniqueness in a command procedure. Issuing UNIQUE from your procedure has no effect (and gives a zero return code) if the current copy of the procedure is the only one active. An *active* long running command or procedure is one that is in any stage of its processing but not yet complete. This includes procedures that are suspended (blocked) by some other long running command. If another copy of the same procedure exists under the same task, UNIQUE has an effect on the entire roll group that includes that copy.

When using UNIQUE with the CANCEL option (the default), the calling procedure is temporarily suspended while the older copy is given control with a reset condition. In this case, the NetView program suppresses the cancellation messages normally issued when a procedure is reset. When the canceled copy of the procedure and any others in its group complete, the issuing copy resumes with the next line after the UNIQUE command. The return code is set to 4.

Using UNIQUE with the PROMOTE option promotes the previous copy of the calling procedure and its roll group to the top of the roll stack, ready to resume when the copy issuing UNIQUE completes. The return code is set to 4 in this case also.

When using UNIQUE in NetView command list language, code a suppression character to suppress unwanted command echoes when the command has an error.

Code SIGNAL ON HALT in your REXX procedures to suppress the REXX cancellation message. The HALT subroutine should return a −5 return code.

Note: No special processing is required for the ROLL command. It is issued exactly like other NetView commands. To be consistent with other NetView applications, set PF6 and PF18 to issue the ROLL command.

Full-Screen Input Capabilities

The VIEW command allows the contents of multiple input-capable variables on a panel, along with the AID information and cursor location, to be returned to the invoking procedure. This is specified with the INPUT keyword and by coding an attribute variable with the FI *type value* pair.

Note: When using the input option, you have an input field only if you defined an attribute variable specifying input. (See "Attribute Variables" on page 16 for information on the *type value* pair.) When the panel is displayed, it contains the variable values that can be modified by typing over them. The modified variables are returned to the invoking procedure when an AID key is pressed. Table 6 describes the AID key. The following variables are set on return to the calling command procedure:

Table 6. Variables Set On Return to Calling Command Procedure

REXX and HLL	NetView Command List Language	Description
VIEWAID	&VIEWAID	The AID key used to enter the input.
VIEWCURROW	&VIEWCURROW	The cursor location (row) when the AID key was pressed.
VIEWCURCOL	&VIEWCURCOL	The cursor location (column) when the AID key was pressed.

The contents of the VIEWAID variable are defined as PF1, PF2, ..., PF24, PA1, PA2, PA3, or Enter.

If PA1, PA2, or PA3 is pressed, only the AID (VIEWAID) information is returned to the invoking procedure. The cursor row and column locations and any input fields defined on a panel are not returned.

The following five figures illustrate source panels using VIEW with the INPUT option to create a rollable component. Figure 2 on page 22 and Figure 3 on page 22 show the source panels containing input-capable variables to be replaced. These panels are using attributes from Attribute Set 2 (see Table 5 on page 15).

```
/**************************************************************************/
/*  FIRST PANEL DISPLAYED                                                 */
/**************************************************************************/
*** AT2
+PANEL1
$ X========================================================================X
$ |                                                                        |
$ |                                                                        |
$ |%     RRRRRRR     AAAAAAA    NN    NN  EEEEEEEE   LL           111   $  |
$ |%     RR    RR   AA      AA  NNN   NN  EE         LL         11 11   $  |
$ |%     RRRRRRRR   AAAAAAAAA   NN NN NN  EEEEEEEE   LL            11   $  |
$ |%     RR         AA      AA  NN   NNN  EE         LL            11   $  |
$ |%     RR         AA      AA  NN    NN  EEEEEEEE   LLLLLLLL   11111111$  |
$ |------------------------------------------------------------------------|
$ |  INPUT VARIABLE 1 = &VARIN1                         %                 $|
$ |  INPUT VARIABLE 2 = &VARIN2                                            |
$ |                                                                        |
$ |  You entered:  &VAROUT1                                                |
$ |  You also entered:  &VAROUT2                                           |
$ X========================================================================X
$
$Enter a command on the command line OR...
$Enter NEXT or press PF8 to view the next panel.
$
%Action==> &COMMAND                                                        %
$                               PF2= End
$           PF6/PF18= Roll      PF8=Next
```

Figure 2. Source for First Panel with Input-capable Variables and Command Line

```
/**************************************************************************/
/*  SECOND PANEL DISPLAYED                                                */
/**************************************************************************/
*** AT2
+PANEL2
$ X========================================================================X
$ |                                                                        |
$ |                                                                        |
$ |%     RRRRRRR     AAAAAAA    NN    NN  EEEEEEEE   LL         22222222 $ |
$ |%     RR    RR   AA      AA  NNN   NN  EE         LL               22 $ |
$ |%     RRRRRRRR   AAAAAAAAA   NN NN NN  EEEEEEEE   LL         22222222 $ |
$ |%     RR         AA      AA  NN   NNN  EE         LL         22       $ |
$ |%     RR         AA      AA  NN    NN  EEEEEEEE   LLLLLLLL   22222222 $ |
$ |                                                                        |
$ |------------------------------------------------------------------------|
$ |                                                                        |
$ |                                                                        |
$ |                                                                        |
$ |                                                                        |
$ |                                                                        |
$ X========================================================================X
$
$Enter a command on the command line OR...
$Enter BACK or press PF7 to view the previous panel.
$
%Action==> &COMMAND                                                        %
$                               PF2= End
$           PF6/PF18= Roll      PF7= Previous
```

Figure 3. Source for Second Panel with Command Line Only

The REXX command procedure shown in Figure 4 invokes VIEW with the INPUT keyword to display PANEL1. It assigns initial values to the VARIN1 and VARIN2 input-capable variables in the source panel. It also returns the AID information and command line input to the caller.

```
/*****************************************************************************/
/*  EXAMPLE:  NETVIEW COMPONENT USING THE VIEW COMMAND                       */
/*****************************************************************************/
SIGNAL ON HALT
/*****************************************************************************/
/*  RESUME OLD COPY IF ONE EXISTS                                            */
/*****************************************************************************/
 'UNIQUE PROMOTE'
 if rc = 4 then EXIT -5              /* -5 will cancel caller if it exists */
SIGNAL ON ERROR             /* any nonzero rc other than as a result of the */
                            /* UNIQUE command is an error                   */
/*****************************************************************************/
/* set up VAR1 and VAR2 as input capable fields                              */
/*****************************************************************************/
$VARIN1 = 'FI IN CR HB UN'
$VARIN2 = 'FI IH CG HR UN'
/*****************************************************************************/
/* set up COMMAND as an input command line using an attribute                */
/* variable.  Also define the cursor to stop at this field.                  */
/*****************************************************************************/
$COMMAND = 'FI UY'
  VARIN1 = 'INITIALIZE 1'
  VARIN2 = 'INITIALIZE 2'
Do forever
  COMMAND = '00'X                        /* COMMAND = nullchar (this clears */
                                         /* the command line and provides   */
                                         /* for insert capability)          */
  'VIEW USERAPPL PANEL1 INPUT'
  UPPER COMMAND
  VAROUT1 = VARIN1
  VAROUT2 = VARIN2
  SELECT
    When viewaid = PF2 then exit                       /* Quit if PF2       */
    When viewaid = PF6 then CMD HIGH ROLL              /* Roll if PF6       */
    When viewaid = PF8 then call PANEL2                /* Next panel if PF8 */
    When viewaid = ENTER then
        SELECT
          when command = NEXT then call PANEL2
          /******************************************************************/
          /*         Assume any other input given on command line is        */
          /*         to be issued to NCCF                                   */
          /******************************************************************/
          when COMMAND ¬= ' ' then
               DO
                'CMD HIGH' COMMAND
               END
          otherwise nop
        END
    OTHERWISE nop
  End                                                        /* select */
End                                                       /* Do forever */
```

Figure 4 (Part 1 of 2). REXX Command Procedure that Drives a Rollable Component

```
PANEL2:
Do forever
  COMMAND = '00'X                           /* COMMAND = nullchar (this clears */
                                            /* the command line and provides   */
                                            /* for insert capability)          */
  'VIEW USERAPPL PANEL2 INPUT'
  UPPER COMMAND
  SELECT
    When viewaid = PF2 then exit                          /* Quit if PF2       */
    When viewaid = PF7 then return                        /* Previous panel PF7*/
    When viewaid = PF6 then 'CMD HIGH ROLL     '                /* Roll if PF6 */
    When viewaid = ENTER then
        SELECT
          When COMMAND = BACK then return
          /*****************************************************************/
          /*          Assume any other input given on command line is      */
          /*          to be issued to NCCF                                 */
          /*****************************************************************/
          when COMMAND ¬= ' ' then
               DO
                'CMD HIGH' COMMAND
               END
          otherwise nop
        END
    OTHERWISE nop
  End    /* select */
End   /* Do forever  */
RETURN
ERROR:
 EXIT -1                      /*  -1 means "FATAL ERROR IN NESTED PROCEDURE" */
HALT:
 EXIT -5                      /*  -5 means "CANCEL REQUESTED"                */
```

Figure 4 (Part 2 of 2). REXX Command Procedure that Drives a Rollable Component

Figure 5 on page 25 is a typical display of this panel. The variables VARIN1 and VARIN2 have been replaced with actual values INITIALIZE 1 and INITIALIZE 2, respectively. (See Figure 2 on page 22 for the source for this panel.) The attribute specification has been defined by $VARIN1 and $VARIN2 (see "Attribute Variables" on page 16 for more information).

VARIN1 attributes are:

Input, tab (unprotected)
Normal intensity
Red
Blinking
No cursor position.

The length of the input field continues until the next attribute symbol is encountered. In this case it is %.

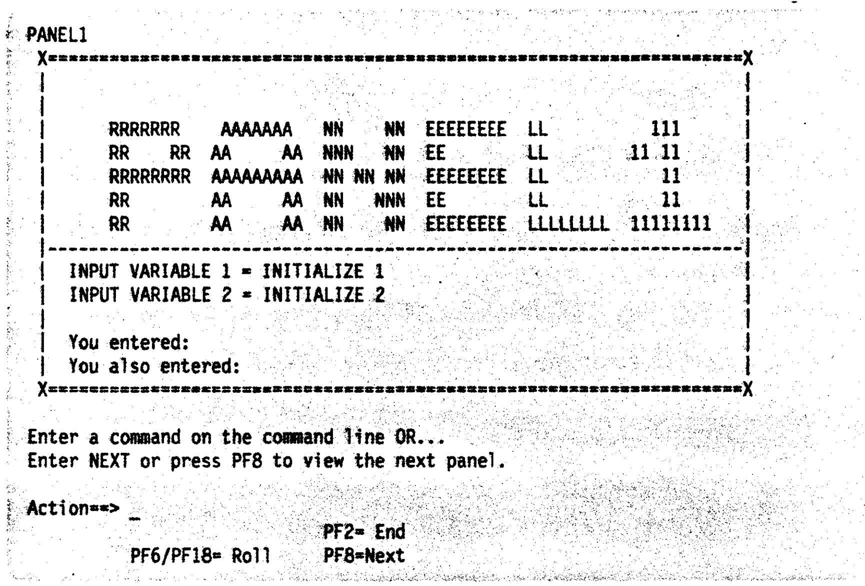
VARIN2 attributes are:

Input, tab (unprotected)
High intensity
Green
Reverse video
No cursor position.

The length of the input field continues until the end of the line.

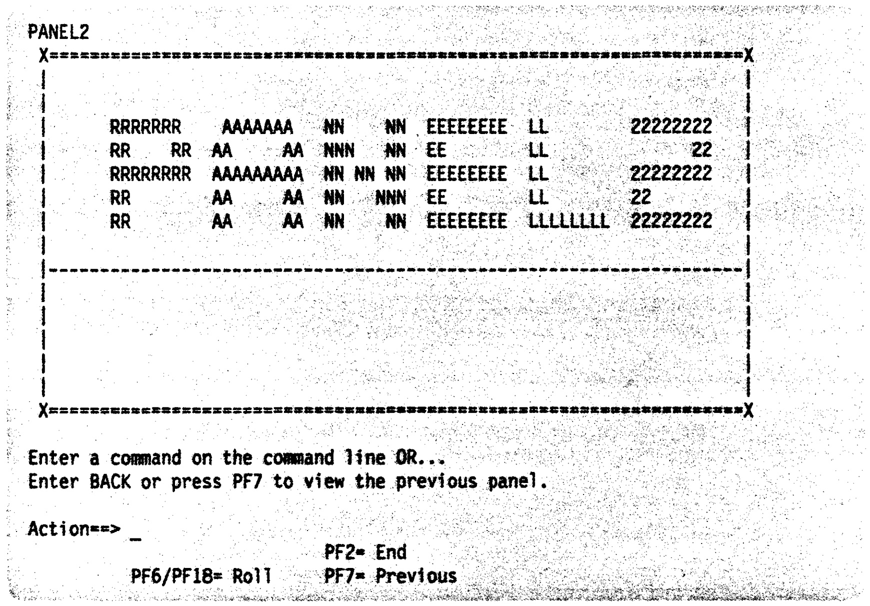
COMMAND attributes are:

Input, tab (unprotected)
Position the cursor at this field.



```
PANEL1
 X=========================================================================X
 |                                                                         |
 |                                                                         |
 |    RRRRRRR    AAAAAAA   NN    NN  EEEEEEEE  LL          111             |
 |    RR    RR  AA      AA  NNN   NN  EE        LL        11 11            |
 |    RRRRRRRR  AAAAAAAAA  NN NN NN  EEEEEEEE  LL           11             |
 |    RR        AA      AA  NN   NNN  EE        LL           11             |
 |    RR        AA      AA  NN    NN  EEEEEEEE  LLLLLLLL  11111111         |
 |-------------------------------------------------------------------------|
 |  INPUT VARIABLE 1 = INITIALIZE 1                                        |
 |  INPUT VARIABLE 2 = INITIALIZE 2                                        |
 |                                                                         |
 |  You entered:                                                           |
 |  You also entered:                                                      |
 X=========================================================================X

Enter a command on the command line OR...
Enter NEXT or press PF8 to view the next panel.

Action==> _
                              PF2= End
          PF6/PF18= Roll      PF8=Next
```

Figure 5. First Display Panel of Component with Variables Replaced by REXX Command Procedure



```
PANEL2
 X=========================================================================X
 |                                                                         |
 |                                                                         |
 |    RRRRRRR    AAAAAAA   NN    NN  EEEEEEEE  LL        22222222          |
 |    RR    RR  AA      AA  NNN   NN  EE        LL              22          |
 |    RRRRRRRR  AAAAAAAAA  NN NN NN  EEEEEEEE  LL        22222222          |
 |    RR        AA      AA  NN   NNN  EE        LL        22                |
 |    RR        AA      AA  NN    NN  EEEEEEEE  LLLLLLLL  22222222         |
 |                                                                         |
 |-------------------------------------------------------------------------|
 |                                                                         |
 |                                                                         |
 |                                                                         |
 |                                                                         |
 |                                                                         |
 X=========================================================================X

Enter a command on the command line OR...
Enter BACK or press PF7 to view the previous panel.

Action==> _
                              PF2= End
          PF6/PF18= Roll      PF7= Previous
```

Figure 6. Second Display Panel of Component

Returning Command Line Input

NOINPUT specified: For the NetView program to process the command line when NOINPUT is specified, you must define a tilde (~) on the panel to be displayed.

The tilde definition defines an input field on the panel that is to be returned to the NetView program as a command. An &CUR coded on this line after the tilde determines where the cursor is to be positioned.

The &CUR is particularly useful for predefining a partial command. For example:

```
~ V NET,ACT,ID=&CUR
```

coded on a panel would display

```
V NET,ACT,ID=_
```

with the remaining ID to be completed by the operator.

Only one command line tilde and only one &CUR are allowed per panel. See Figure 10 on page 36 for the PF keys provided by the NetView program when NOINPUT is specified.

INPUT specified: When INPUT is specified, the command line should be coded the same as any other input-capable field. The &CUR and tilde definitions should not be used. It is the responsibility of the procedure that is displaying the panel to issue any commands. See "Issuing Commands from Command Procedures" on page 19 for information on issuing CMD HIGH.

Dynamic Update Capabilities — MSG Option

The VIEW command enables you to dynamically update the contents of the panel being displayed. All panel variables that might be updated by message automation should be declared as global variables in a NetView command list or HLL command procedure that uses the VIEW command. VIEW displays a panel, and the contents of the global variables are substituted. The incoming messages may dynamically update some of the global variables currently displayed on the panel. If MSG is specified, a displayed panel is interrupted by *any* message received by the OST under which VIEW is running. The contents of the displayed panel can then be refreshed without operator intervention. When NOMSG is specified, incoming messages do not interrupt a displayed panel.

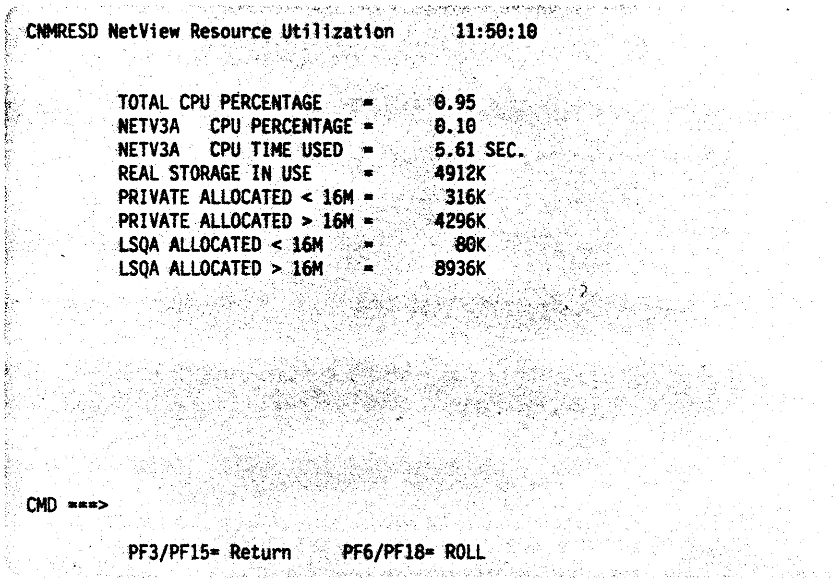
You can code a command procedure to invoke VIEW with the MSG option. Code a message automation table entry to trap the message and execute a second command procedure to update the variables. The first command procedure declares global variables and invokes VIEW. VIEW displays a panel substituting the contents of the global variables declared in the command procedure. When VIEW is interrupted by a message received at the OST, the incoming message drives the second command procedure that has updated some of the global variables currently displayed on the panel. The interrupted panel is redisplayed with the new values of the global variables changed by the second command procedure.

Sample of Panel Updating

The following four figures illustrate the dynamic update of the contents of a panel.

Figure 9 on page 29, command list RESDYN, uses the RESOURCE command output to illustrate how to use the MSG operand of the VIEW command. The data displayed is updated on a time interval that you specify when invoking the command list (RESDYN); the default time interval is 10 seconds.

Figure 7 shows a typical RESDYN command.

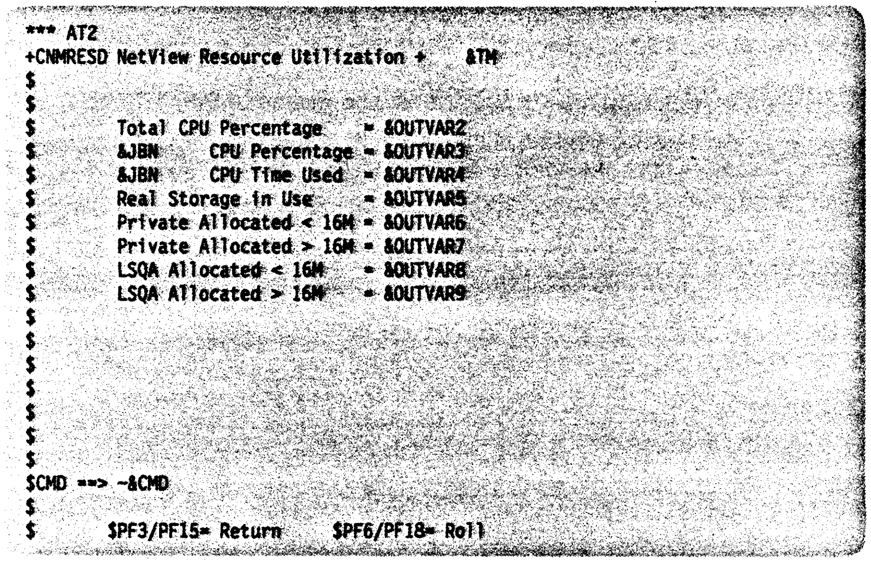


```
CNMRESD NetView Resource Utilization      11:50:10


          TOTAL CPU PERCENTAGE     =      0.95
          NETV3A   CPU PERCENTAGE =       0.10
          NETV3A   CPU TIME USED  =       5.61 SEC.
          REAL STORAGE IN USE      =      4912K
          PRIVATE ALLOCATED < 16M =        316K
          PRIVATE ALLOCATED > 16M =       4296K
          LSQA ALLOCATED < 16M     =        80K
          LSQA ALLOCATED > 16M     =      8936K










CMD ===>

            PF3/PF15= Return    PF6/PF18= ROLL
```

Figure 7. RESDYN Command Output Example

RESDYN uses the MSG option so that the commands scheduled by EVERY can change global variables displayed and allow the panel to be automatically refreshed. You also specify the MSG option if updates were scheduled by message automation. Any other messages the operator receives are also processed. When RESDYN is scheduled by EVERY processing, it updates task global variables. Invoking the original VIEW automatically picks up the new values when it resumes.

Figure 8 is the source panel text that displays the previous panel (Figure 7).



```
*** AT2
+CNMRESD NetView Resource Utilization +    &TM
$
$
$        Total CPU Percentage     = &OUTVAR2
$        &JBN     CPU Percentage = &OUTVAR3
$        &JBN     CPU Time Used  = &OUTVAR4
$        Real Storage in Use     = &OUTVAR5
$        Private Allocated < 16M = &OUTVAR6
$        Private Allocated > 16M = &OUTVAR7
$        LSQA Allocated < 16M    = &OUTVAR8
$        LSQA Allocated > 16M    = &OUTVAR9
$
$
$
$
$
$
$
$CMD ==> ~&CMD
$
$       $PF3/PF15= Return     $PF6/PF18= Roll
```

Figure 8. CNMRESD Source Panel

VIEW manages the PF keys and the command line without intervention by the RESDYN command list.

Figure 9 shows the results of using VIEW in a command procedure (using the RESDYN command list) to display the results of the RESOURCE command on a full-screen panel. It also shows the use of VIEW by a command procedure.

```
/*********************************************************************************/
/*                                                                               */
/*  Display the results of the RESOURCE command on a full screen panel.         */
/*      Syntax:  RESDYN  interval                                                */
/*  "interval" is the number of seconds (from 1 to 59) between updates.        */
/*                                                                               */
/*  RESDYN runs in two modes: When invoked by an operator, it                   */
/*  initializes, then issues VIEW.  It also runs as a result of                  */
/*  EVERY scheduling, for the purpose of updating the (global)                  */
/*  variables with the data to be displayed.  The original VIEW                 */
/*  invocation automatically picks up the latest values whenever                */
/*  it is resumed.                                                               */
/*                                                                               */
/*********************************************************************************/
 SIGNAL ON HALT                                /* Always used with VIEW         */
 SELECT                                        /* How were we driven?           */
  WHEN  msgvar(1) = 'UPD' THEN                 /* For update?  (from EVERY)     */
   DO
     CALL  TRAPRTN                             /* Yes, update global variables */
     EXIT                                      /*  that's all!                  */
   END
  WHEN  msgvar(1) = ''  THEN                   /* By operator, with default?   */
     timev = 10;                               /*  Yes, default is 10 seconds  */
  WHEN  DATATYPE(msgvar(1)) = 'NUM' THEN       /* By operator w/value?          */
   DO                                          /* Yes,                          */
     IF  msgvar(1)<3 | msgvar(1)>59  THEN      /* value acceptable?             */
        SIGNAL PARMERROR                       /*  No, tell OP bad news.        */
     timev = right(msgvar(1),2,'0')            /*  EVERY command needs 2 digits*/
   END
  OTHERWISE  SIGNAL PARMERROR                  /* Any other way is bad.         */
 END

 CALL TRAPRTN                                  /* Initialize global vars        */
```

Figure 9 (Part 1 of 3). REXX Command List to Invoke RESOURCE Command Processor

```
 'TRAP AND SUPPRESS MESSAGES DSI208I'         /* trap output from EVERY    */

/*****************************************************************************/
/* When using VIEW with the NOINPUT option, UNIQUE is not ordinarily        */
/* required.  However, in this case, we want to be sure that there          */
/* is not already a copy executing before issuing the EVERY command,        */
/* which would fail, otherwise.                                             */
/*****************************************************************************/
 'UNIQUE'
            /* at this point previous copy (if any) has completed.          */
 'EVERY 00:00:'timev',ID=RESDYN, RESDYN UPD'   /* Schedule updates          */
 IF  rc ¬= 0  THEN  SIGNAL SCHDERROR
/*****************************************************************************/
/*  Define display attributes for the variables in the panel.              */
/*  $OUTVAR2 is a global because it is changed by update processing.       */
/*  These variables do not need to be task globals because they            */
/*  are not updated when RESDYN is called for update.                      */
/* IH = high intensity                                                      */
/* CW = white   CR = red                                                    */
/*****************************************************************************/

$OUTVAR3 = 'IH CW'
$OUTVAR4 = 'IH CW'
$OUTVAR5 = 'IH CW'
$OUTVAR6 = 'IH CW'
$OUTVAR7 = 'IH CW'
$OUTVAR8 = 'IH CW'
$OUTVAR9 = 'IH CW'
$TM      = 'IH CP'

/*****************************************************************************/
/*  Display a panel with the current values of the global variables.       */
/*  VIEW will automatically update the fields being shown whenever         */
/*  the values of the global variables are changed.                        */
/*                                                                          */
/*****************************************************************************/
'VIEW RESDYN CNMRESD MSG'
IF  rc¬=0  THEN  'SHOWCODE' rc 'CNMRESD'
"PURGE TIMER=RESDYN"
EXIT
```

Figure 9 (Part 2 of 3). REXX Command List to Invoke RESOURCE Command Processor

```
/*********************************************************************************/
/* This subroutine extracts information from the message produced by           */
/* the RESOURCE command and sets task global variables.                          */
/*********************************************************************************/
TRAPRTN:
  'TRAP AND SUPPRESS MESSAGES DSI386I'   /* trap output from RESOURCE         */
  'RESOURCE'
  IF  rc ¬= 0  THEN                    /* RESOURCE failed? under PPT?         */
   DO
     say "RESDYN's RESOURCE command failed with" rc
     "UNIQUE CANCEL"                   /* Stop the 'main' copy for this error. */
     EXIT                              /* Stop this one, too.                 */
   END

                                       /* 'WAIT FOR MESSAGES' not necessary   */
                                       /* since message from RESOURCE are     */
                                       /* synchronous.                        */

  'MSGREAD'
  IF  rc ¬=0  THEN
   DO
     say "RESDYN's message read failed with" rc
     EXIT
   END

  DO  cntr = 2 to 9
     'GETMLINE LINE'cntr cntr
     'PARSEL2R LINE'cntr '/=/ RESULT'
     INTERPRET 'OUTVAR'cntr ' = VALUE("RESULT")'
     IF cntr=3  THEN
        'PARSEL2R LINE'cntr 'JBN P1' /* Get the jobname                       */
  END
  TM = TIME()
  IF  OUTVAR2 > 50  THEN               /* CPU utilization is high?            */
   $OUTVAR2 = 'IH CR HR'               /* High intensity, red, reverse        */
  ELSE
   $OUTVAR2 = 'IH CW'                  /* High intensity, white               */

  'GLOBALV PUTT OUTVAR2,OUTVAR3,OUTVAR4,OUTVAR5,',
     'OUTVAR6,OUTVAR7,OUTVAR8,OUTVAR9,TM'

  'GLOBALV PUTT $OUTVAR2'              /* This one changes, must be global.   */
  RETURN

PARMERROR:                             /* Operator specified invalid time     */
                                       /* interval.                           */
  say 'Invalid parameter' msgvar(1) 'for RESDYN, specify number'
  say 'between 3 and 59'
  exit 12

SCHDERROR:                             /* EVERY command failed, shouldn't     */
                                       /* happen.
  say 'RESDYN cannot schedule updates. EVERY command failed with' rc
  exit 16

HALT:                                  /* RESDYN has been reset,              */
                                       /* possibly by UNIQUE command

 "PURGE TIMER=RESDYN"
  EXIT -5                              /* Requesting caller,                  */
                                       /* if any, to cancel, also.
```

Figure 9 (Part 3 of 3). REXX Command List to Invoke RESOURCE Command Processor

Return Codes from VIEW

Table 7. Return Codes from VIEW

Code	Meaning	Your Action
4	Panel specified not found in CNMPNL1 or CNMMSGF data sets (MVS).	Put panel definition in correct data set or file.
8	Panel definition format not valid; no noncomment lines found.	Correct format of panel definition.
16	VIEW command processor invoked with invalid parameters. *Name1* must be 1 to 8 characters and *name2* must be valid panel name. Valid parameters are INPUT, NOINPUT, MSG, NOMSG.	Correct command list to use valid option.
24	Full-screen command processor is available to OST only.	Do not invoke VIEW from a non-OST.
28	Logical record length of panel not 80 bytes.	Change data set or file to logical record length of 80 bytes.
32	Unrecoverable error resulted from macro call. For VM, error could be that panel specified was not a file with filetype NCCFLST.	Call IBM for service.
36	Unrecoverable internal programming error occurred.	Call IBM for service.
81	Panel definition format not valid; no text indicator line found, or more than 49 option definitions found. (See page 36.)	Correct format of panel definition.
82	Panel definition format not valid; panel data stream exceeded 9600 bytes.	Correct format of panel definition.
83	Panel definition format not valid; comment lines in wrong place.	Correct format of panel definition.
84	Panel definition format not valid; more than 24 displayable lines found.	Correct format of panel definition.
85	Panel definition format not valid; more than one command line defined, or more than one &CUR defined in command line when VIEW invoked with NOINPUT specified.	Correct format of panel definition.

Displaying VIEW Return Codes

The SHOWCODE command list is used by command procedures to display descriptions of the return codes from the VIEW command. The descriptions are displayed in the form of messages. You can use SHOWCODE for the same purpose when you create or customize your command procedures.

These are the valid return codes and their related message IDs:

4	CNM335I
8	CNM336I
12	CNM337I
16	CNM338I
24	CNM340I
28	CNM341I
32	CNM342I
36	CNM343I
81 − 85	CNM336I

See Figure 9 on page 29 for an example using SHOWCODE to display error messages from VIEW.

You can code similar functions into command procedures you create or customize. See *NetView Customization: Writing Command Lists* for more information.

Chapter 3. Modifying and Creating Online Panel Interfaces

This chapter tells how to change the content of the online help facility panels in the NetView program. Online help uses the VIEW command which was described in the previous chapter. To use the PF keys, menu selection, and panel hierarchies described in this chapter, you must specify the NOINPUT option when you invoke the VIEW command. This chapter also tells how to create new panels for your own help applications.

Note: If your panels or alert messages have been translated into Japanese (using Kanji), exercise great care to preserve the integrity of the double byte character set strings.

Modifying Help Panels

Reasons for modifying an online help panel include changing panel text and adding an item to a menu. The general procedure for modifying a displayed panel is to:

1. Locate its source panel
2. Copy the source panel
3. Type your changes into the copy of the source panel
4. Store the modified panel
5. Test the modified panel.

Locating a Source Panel

The first step in modifying an online help panel is to determine the source panel that controls the panel you want to change.

A *source panel* is a file that defines the contents of the *displayed panel* that appears on the end user's screen. A source panel contains the text of the displayed panel as well as definition statements that do not appear on the displayed panel. These definition statements include a prolog, the name of the help panel for the particular displayed panel, and a list of panels the operator can select to appear next.

Source panels are stored in a data set concatenated to CNMPNL1 (for MVS) or in a file with filetype NCCFLST (for VM). For message help panels, the source is a VSAM file, CNMMSGF (for MVS and VM).

To view the source panel for a particular displayed panel, enter:

```
BROWSE panel_name
```

where *panel_name* is the name that appears in the upper left corner of the displayed panel.

Modifying a Source Panel

To modify a source help panel, use a screen editor such as ISPF/PDF (for MVS) or XEDIT (for VM). Make your changes to the source panel by typing over the existing text or by adding text. For message help panels, see "Modifying Online Message Panels" on page 39. The items of text you can change are numbered in the sample source panel shown in Figure 10 on page 36. The text that follows the figure describes these fields.

Note: All panels must have a logical record length of 80 bytes. Null characters are also counted within this 80-byte record.

```
/* This is the main menu for the NetView help desk. 1
/*
/*
HELP=CNMHHINT  HOW TO USE THE HELP DESK 2
1 CNMHTI      THE TERMINAL DOES NOT WORK
2 CNMHA1      APPLICATION PROBLEMS
3 CNMHP1      PERFORMANCE PROBLEMS 3
4 CNMHH1      PROBLEMS IDENTIFIED THROUGH NETWORK MONITORING
5 CNMHM1      SYSTEM MESSAGE CROSS-REFERENCE
*** 4
+CNMHDESK 6                 %NETVIEW HELP DESK 7
$
$
\Select+      To get information about
$
$  %1    $A terminal not working
$
$  %2    $A transaction or an application not working
$                                                                       5
$  %3    $Slow response time
$                                                              8
$  %4    $Problems identified through network monitoring
$
$  %5    $System message cross-reference
$
+Type a number (1 through 5) and press ENTER.
$
%       PF1 --->+Recommendations for Setup and Use of the help desk.
$
%                     HELP NETVIEW ---> $NetView Help Menu
$ 9
%Action===>~&CUR 10
+ 11        %PF1= Help  $PF2= End   PF3= Return   PF4= Top  PF5= Bottom
+          $PF6= Roll   PF7= Backward   PF8= Forward   $PF11= Entry Point
```

Figure 10. Sample Source Panel

(The special characters you see in the source panel, such as the dollar sign ($) and the percent sign (%), are described under "Controlling Color and Highlighting of Fields" on page 14.)

1 Prolog	An optional section for programmer comments. Each line of the prolog begins with /* in columns 1 and 2.
2 Help	Optional text that defines the help panel (helppan) for a particular displayed panel. This field follows the prolog and is coded in this format:

```
Column
1             15
HELP=helppan  comment
```

See "Naming a Source Panel" on page 40 for naming conventions.

PF1/PF13 brings the specified help panel to the screen; PF3/PF15 brings the originating panel back. If the panel is continued on a second page, its continuation page is coded as described under 3.

3 Option Definitions Optional list of panels from which the operator can select the next panel to be displayed. This list may not exceed 49 option definitions. The list is coded in this format:

```
Column
1 3        12
n panname  comment
```

where *n* is the alphameric character by which the operator calls the panel. To produce a continuation panel, *n* is blank, as shown in the following example:

```
Column
1 3         12
  panname   comment
```

In this case, *panname* identifies the continuation panel. See "Naming a Source Panel" on page 40 for naming conventions.

4 Text Indicator	Three required asterisks that separate the prolog, help, and option definitions from the displayed panel text. The three asterisks followed by a space and then the AT2 option (*** AT2 or "*** KK AT2" for Katakana) indicate that Attribute Set 2 is to be used for color and highlighting. (For Attribute Set 1, use ***, or *** AT1. For Katakana, use "*** KK" or "*** KK AT1".) See "Controlling Color and Highlighting of Fields" on page 14 for more information on Attributes Sets 1 and 2.
5 Panel Text	Up to 24 lines of text that constitute the displayed panel. Command list variables may appear anywhere in the panel text. See "Displaying Variables in Source Panels" on page 17 for more information.
6 Name	Name of the panel. See "Naming a Source Panel" on page 40 for naming conventions.
7 Heading	Text that describes the use of the panel.
8 Text	Body of panel up to 18 lines.
9 Message Area	The 21st line of the panel, on which NetView messages appear.
10 Command Line	The 22nd line of the panel, on which NetView operators type commands to the NetView program. In a VIEW command with the NOINPUT option specified (see Chapter 2, "Using the VIEW Command" on page 13), at least one command line is defined by the ~ attribute symbol. The word *Action* is to the left of the command line. &CUR identifies the cursor position within the command line. Only one input field and only one &CUR are permitted per panel. &CUR is particularly useful for predefining a command in the input field. Without &CUR, the cursor defaults to the first position in the input field.

11 PF Key Prompts The 23rd and 24th lines of the panel indicate the function of specific PF keys:

PF1 = HELP	Displays the panel coded as HELP=helppan.
PF2 = END	Exits to the originating component.
PF3 = RETURN	Returns to the last panel from which a selection was made.
PF4 = TOP	Displays the first page when you are in a multipage panel series.
PF5 = BOTTOM	Displays the last page when you are in a multipage panel series.
PF6 = ROLL	Rolls to the screen of the previous active component. An operator can use ROLL to see the current panel in all active NetView components.
PF7 = BACKWARD	Returns to the previous page or panel in sequence.
PF8 = FORWARD	Displays the next page when you are in a multipage panel series.
PF11 = ENTRY POINT	Shows the panel that the operator first saw upon entry to Help.

Online Message Panel Naming Conventions

The first message panel of a series of panels, or a message panel that has only one panel, is called the primary panel. The following conventions must be followed when naming an online message panel. Panel naming errors are returned when storing a panel using CNMSJ018 for MVS or CNMSV018 for VM if these conventions are not followed. The identifier for primary panels is the message ID with the action suffix omitted. For example, message ID DSI000I would have the primary panel named DSI000. Primary panel naming conventions are:

- All names must contain six or seven characters
- Characters 1 through 3 are alphabetic
- Characters 4 through 6 (or 7) are numeric.

For example,

```
    DSI000, BNJ1353, ABC1234
```

When one panel cannot contain all the information required, you can use a continuation panel. The continuation panel name is the primary panel's name with an alphabetic suffix, A to X only. For example, DSI000A is the second in a series of panels for message ID DSI000.

Note: The maximum number of panels in a panel hierarchy is 25.

Before a continuation panel can be stored, a primary panel must exist. For example, panel DSI000 must exist before DSI000A can be added. An attempt to store DSI000B before DSI000A results in an error from CNMSJ018 (for MVS) or CNMSV018 (for VM).

Modifying Online Message Panels

The technique for modifying online message panels is similar to regular help panels. However, you must first copy the panel record from the VSAM file (CNMMSGF) to a QSAM file. To do this:

- **For MVS**:
  - Use an editor such as ISPF/PDF to edit the CNMSJ017 sample.
  - Make sure the STEPLIB contains the proper data set name.
  - Enter the panel ID you want to copy into the SYSIN statement. Only one panel can be copied at a time. For example, enter DSI000A into the SYSIN statement to copy the second panel for message DSI000.
  - Submit the job.
  - When the copy is made, the panel requested is placed in a sequential data set named CNM.PANEL.
  - Error messages and successful completion messages are printed unless the JCL has been modified to redirect the output. Check the output designated by the JCL for error messages.
- **For VM**:
  - Invoke CNMSV017 with the panel ID to be copied as a parameter. For example, to copy the second panel for message DSI000 enter:

    ```
    CNMSV017 DSI000A
    ```
  - When the copy is made, the panel requested is placed in a file named *panelid* PANEL.
  - Error messages and successful completion messages are printed unless the exec has been modified to redirect the output.

    ```
    CNMSV017
    ```

    has been modified to redirect the output. Check the output designated by the

    ```
    CNMSV017
    ```

    or error messages.

Note: Any information following the panel identifier is considered a comment and is ignored during processing.

When you have made your changes to the source panel, turn to "Storing New and Modified Panels" on page 43 and "Testing Newly Stored Panels" on page 44 to complete the modification procedure.

Creating New Panels

You may want to create a new panel to provide information for a new menu item, to add information to an existing flow of panels, or to have panel variables assigned by a command list and replaced with actual values. Another reason may be to provide help for a user-defined code point or for a command you have renamed with the CMDSYN parameter. See *NetView Administration Reference* for a description of the CMDSYN statement.

The general procedure for creating a new panel is to copy or create the source panel for it, type your data into the source panel, store it, and test it.

Note: For MVS, if you want to modify or create a panel while the NetView program is running, define your panel data set without secondary extents. Otherwise, a panel might be filed in a new extent, and you would have to close and restart the NetView program to use the panel. For VM, if you want to modify or create a panel while the NetView program is running, you may need to reaccess the minidisk from the NetView USERID.

Creating a Source Panel

Before you create an entirely new source panel, try to locate an existing online help panel that is similar to the panel you want to create. Use the BROWSE command to search for such a panel. If you find one, copy it using a screen editor.

If you have renamed a command, copy the panel that defined the original command.

If you cannot find a similar online help panel, use a screen editor to build the source panel. Refer to Figure 10 on page 36 as a guide for creating the source panel.

Naming a Source Panel

Certain naming conventions apply to source panels that you create, and these must be adhered to. For message panels only, see "Modifying Online Message Panels" on page 39. Make sure your panel names do not conflict with IBM-supplied panel names.

For any new panel, add one or more *R*s to the panel name displayed on the screen (on the first line of the panel definition), so that the displayed name has at least nine characters. Since existing online help panel names have eight characters, the 9-character name will help you identify panels you have created.

If the HELP command list is to access the new panel, the first four letters of the panel name must be CNMK. Determine the remainder of the panel name as follows:

- If the HELP command list is to process one operand, the first two and last two characters of the operand should follow CNMK. For example, the name for a help panel displayed for HELP APPLSACT would be CNMKAPCT.
- If the HELP command list is to process two operands, the first two characters of each operand should follow CNMK. For example, the name for a help panel displayed for HELP NETVIEW COMMANDS would be CNMKNECO.

If the new panel is to replace an existing panel, the name of the new file must match the name of the existing file.

If the new panel is to provide help for a renamed command, change the copied panel's name to reflect the new command name. Define the new name for the copied panel with the CMDSYN resource definition statement.

Writing a Source Panel

The instructions for structuring a new panel are the same as those for modifying an existing panel (see "Modifying a Source Panel" on page 35). In addition, you might need to change a command list or another panel that is affected by your new panel. Depending on how the new panel will be used, refer to one of the following sections:

- If the new panel is to display variables assigned by a command list, see "Displaying Variables in Source Panels" on page 17.

- If the new panel is to be called by a command list, see Chapter 2, "Using the VIEW Command" on page 13.
- If the new panel is to be a new menu option, see "Adding a Menu Item."
- If the new panel is to be linked into a flow of two existing panels, see "Linking into an Existing Flow" on page 42.

After you have written the new panel and made any other necessary changes, follow the instructions under "Storing New and Modified Panels" on page 43 and "Testing Newly Stored Panels" on page 44.

Note: When coding Kanji on a help panel, you must enclose the Kanji characters with the shift-out (X'0E') and the shift-in (X'0F') on the same line. If the panel has more than one line containing Kanji text, code both the shift-out and shift-in on each line.

Adding a Menu Item

If the panel you create is to be a new menu item, add the name and description of the new panel to the source panel for the menu. For example, suppose Figure 11 is the existing panel to which you want to add a new menu option, V.

```
/*
/*
/*
HELP=CNM5HCCL
A CNMKAPLS APPLS Command List
B CNMKAPCT
C CNMKAPEN
D CNMKBFSE BFRUSE Command List
E CNMKCDMS CDRM Command List
F CNMKCDCS CDRSCS Command List
G CNMKCLRS
H CNMKDIIS
I CNMKDISG
J CNMKDRTE
K CNMKERST
L CNMKLIES
M CNMKMAES
N CNMKNCOR
O CNMKNODE
P CNMKPAHS
Q CNMKPENG
R CNMKSTNS
S CNMKTEMS
T CNMKTSER
U CNMKVRST
***
+CNMOCDMU           %VTAM COMMAND LISTS TO DISPLAY NETWORK RESOURCES
$
$
$\Select+                                  \Select+
$
$   %A  $APPLS   +applications                %L  $LINES   +lines
$   %B  $APPLSACT+active applications         %M  $MAJNODES+major nodes
$   %C  $APPLSPEN+pending applications        %N  $NCPSTOR +NCP storage
$   %D  $BFRUSE  +buffer use                  %O  $NODE    +any node
$
$   %E  $CDRMS   +cross-domain resource mgrs %P  $PATHS   +paths
$   %F  $CDRSCS  +cross-domain resources     %Q  $PENDING +pending nodes
$   %G  $CLSTRS  +clusters                   %R  $STATIONS+link stations
$
$   %H  $DIS     +display any resource       %S  $TERMS   +terminals
$   %I  $DISG    +display route graphics     %T  $TSOUSER +TSO users (MVS only)
$   %J  $DROUTE  +routes                     %U  $VRST    +virtual route status
$   %K  $ERST    +explicit route status
$
+Type a letter (A through U) and press ENTER.
$
%Action===>~&CUR
$          PF1= Help   PF2= End   PF3= Return   PF4= Top   PF5= Bottom
$          PF6= Roll   PF7= Backward   PF8= Forward   $PF11= Entry Point
```

Figure 11. Source Panel with a Menu

To make your new panel the last menu option in Figure 11, do the following (the referenced additions are reflected in Figure 12 on page 42):

1 Add an option definition to the list of options.
2 Add an *R* to the name of the panel displayed on the screen.
3 Add the panel name, letter key, and description to the menu selections.
4 Add the letter key V to the action options.

Make adjustments to maintain 24 or fewer lines of displayable text.

```
/*
/*
/*
HELP=CNM5HCCL
A CNMKAPLS APPLS Command List
B CNMKAPCT
C CNMKAPEN
D CNMKBFSE BFRUSE Command List
E CNMKCDMS CDRM Command List
F CNMKCDCS CDRSCS Command List
G CNMKCLRS
H CNMKDIIS
I CNMKDISG
J CNMKDRTE
K CNMKERST
L CNMKLIES
M CNMKMAES
N CNMKNCOR
O CNMKNODE
P CNMKPAHS
Q CNMKPENG
R CNMKSTNS
S CNMKTEMS
T CNMKTSER
U CNMKVRST
V CNMKVTER 1
***
+CNMOCDMUR 2          %VTAM COMMAND LISTS TO DISPLAY NETWORK RESOURCES
$
$\Select+                               \Select+
$
$   %A  $APPLS    +applications              %L  $LINES    +lines
$   %B  $APPLSACT+active applications        %M  $MAJNODES+major nodes
$   %C  $APPLSPEN+pending applications       %N  $NCPSTOR +NCP storage
$   %D  $BFRUSE   +buffer use                %O  $NODE     +any node
$
$   %E  $CDRMS    +cross-domain resource mgrs %P  $PATHS    +paths
$   %F  $CDRSCS   +cross-domain resources    %Q  $PENDING +pending nodes
$   %G  $CLSTRS   +clusters                  %R  $STATIONS+link stations
$
$   %H  $DIS      +display any resource      %S  $TERMS    +terminals
$   %I  $DISG     +display route graphics    %T  $TSOUSER +TSO users (MVS only)
$   %J  $DROUTE   +routes                    %U  $VRST     +virtual route status
$   %K  $ERST     +explicit route status     %V  $VTAMUSER+user Command List 3
$
+Type a letter (A through V) and press ENTER. 4
$
%Action===>~&CUR
$          PF1= Help   PF2= End   PF3= Return   PF4= Top   PF5= Bottom
$          PF6= Roll   PF7= Backward   PF8= Forward   $PF11= Entry Point
```

Figure 12. Source Panel with a New Menu Item

Create the panel for this new item and name it CNMKVTER. Then save the altered menu with the name CNMOCDMU.

Linking into an Existing Flow

To insert a newly created panel into a flow of two existing panels, modify the list of options in the top left corner of the source panels. For example, if panels A and B are part of an existing flow and C is a new panel that belongs in the flow, add panel C to source panel B's list of panel options.

Figure 13 on page 43 represents the existing flow of two panels, and Figure 14 on page 43 represents the revised flow after the addition of a new panel. In the figures, CNMKDRTE represents panel **A**, CNMOCDR1 represents panel **B**, and CMNOCDR2 represents panel **C**.

A

```
CNMOCDR1  DROUTE COMMAND LIST SAMPLE
***
+CNMKDRTE              %DROUTE COMMAND LIST DESCRIPTION       +Page 1 of 2
$
```

B

```
***
+CNMOCDR1                %DROUTE COMMAND LIST SAMPLE            +Page 2 of 2
$
```

Figure 13. Action Options of Panels in an Existing Flow

Figure 14 shows the revised action options of each panel in the new flow after the addition of the new panel.

A

```
CNMOCDR1   DROUTE COMMAND LIST SAMPLE
***
+CNMKDRTER          %DROUTE COMMAND LIST DESCRIPTION         +Page 1 of 3
$
```

B

```
CNMOCDR2   DROUTE COMMAND LIST SAMPLE CONT.
***
+CNMOCDR1R            %DROUTE COMMAND LIST SAMPLE              +Page 2 of 3
$
```

C

```
***
+CNMOCDR2R           %DROUTE COMMAND LIST SAMPLE CONT.         +Page 3 of 3
$
```

Figure 14. Action Options of Panels in a New Flow

In Figure 14, notice how the option definitions have been rearranged. The option definition of the first panel still points to the second panel. But the option definition of the second panel now points to a new continuation panel. This new continuation panel has no option definitions since it is the last panel in the sequence.

Two other changes are necessary. Change the page numbers, as shown in Figure 14, to reflect three pages instead of two. Change the panel names in the upper left corner of the displayed panel by adding as many *R*s to the end of the name as necessary to make the displayed name nine characters long. This distinguishes the modified panels from the unmodified panels.

Storing New and Modified Panels

Store the panels you modify or create in the data set concatenated to the front of DDNAME CNMPNL1 (for MVS) or in a file with filetype NCCFLST (for VM). Panel data streams are limited to 9600 bytes.

For online message panels, you can either store them as described above or add or replace panels directly to the VSAM file (CNMMSGF). To store panels directly on the VSAM file:

- **For MVS**:
  - Edit the CNMSJ018 sample.
  - Make sure the STEPLIB contains the proper data set name.
  - Make sure the CNMSEQ DD statement specifies the proper sequential data set name.
  - Enter the panel ID you want to store into the SYSIN statement. Only one panel can be stored at a time. For example, enter DSI000A into the SYSIN statement to store the second panel for message DSI000.
  - Submit the job.
  - When the store is successful, a message is generated indicating successful completion.
  - Error messages and successful completion messages are printed unless the JCL has been modified to redirect the output. Check the output designated by the JCL for error messages.
- **For VM**:
  - The panel to be stored must have a filename the same as the panel name, have a filetype of "panel," and exist on the A-disk.
  - The STATMON LOADLIB must be accessible.
  - The VSAM DOS disk must be linked in read/write mode. The default device address is 198. If this is not true for your installation, you should modify CNMSV018 to reflect the correct device address.
  - Invoke CNMSV018 with the panel ID to be stored as a parameter. For example, to store the second panel for message DSI000 enter:

    ```
    CNMSV018 DSI000A
    ```
  - When the store is successful, a message is generated indicating successful completion.
  - Error messages and successful completion messages are printed unless the exec has been modified to redirect the output. Check the output designated by the exec for error messages.

Any information following the panel identifier is considered a comment and is ignored during processing. Panel naming conventions can be found in "Modifying Online Message Panels" on page 39. We suggest you retain copies of any message help panels that you add or modify.

Testing Newly Stored Panels

After storing modified or new panels, call each panel and verify that it displays the proper information. To view the panel, enter:

```
TUTOR panel_name
```

where *panel_name* is the name of the appropriate source panel.

If the displayed panel contains errors, modify it, store it, and test it again. If a panel is missing, a CNM*xxx*I message is displayed with a return code and an explanation of that return code. Refer to Table 7 on page 32.

Chapter 4. Customizing Hardware Monitor Displayed Data

This chapter describes how to modify generic and pregeneric alerts. In previous releases of the NetView program, Recommended Action panels, Event Detail panels, and alert messages were stored at the host. Each pregeneric alert had a unique set of panels and messages. With generic alerts, coded information is contained in the alert and this information (generic alert code points) is used to dynamically build the hardware monitor panels.

This chapter tells how to:

- Modify the text of pregeneric Recommended Action and Event Detail panels
- Modify pregeneric alert messages
- Overlay Recommended Action numbers from a generic alert
- How to control the use of color and highlighting for hardware monitor panels
- Use NMVT support for user-written programs including:
  - a sample generic alert and its associated hardware monitor panels
  - creating user-defined generic code points
  - adding additional resource types to the hardware monitor.

Note: If your panels or alert messages have been translated into Japanese (using Kanji), exercise great care to preserve the integrity of the double byte character set strings.

Modifying Hardware Monitor Panels

Recommended Action panels and Event Detail panels are defined for event conditions that are not based on generic alert records. If several event conditions use the same Recommended Action panel or Event Detail panel, the panel is physically defined under a single name, the *actual panel name*. Any other name under which the actual panel may be displayed is the *panel alias*. Determining whether the panel name is an actual name or an alias is the first step in modifying panel text.

- Changes you make to an actual panel text (and name) are reflected in all its aliases.
- Changes you make to a panel alias (or name) cause a new actual panel (and name) to be created under the name that was formerly the alias name.

Determining Panel Name, Actual versus Alias

To determine a panel's name and whether it is an actual panel name or an alias, follow these steps:

You must first determine the event associated with the text you want to change, then identify a resource for which the event has been logged. For example,

1. For the identified resource, display the Alerts-Static, Alerts-History, or Most Recent Events panel.
2. Enter `sel# C`, where *sel#* is the selection number on the panel of the event associated with the text you want to change.
3. Examine the 5-digit code, *xxxyy*, that the NetView program returns.

   xxx is the NetView-designated product code, or block ID, for the resource. (Block IDS are listed in *NetView Problem Determination and Diagnosis*.)
   yy is an individual panel identifier.

Note: If you get a PRODUCT ID and ALERT ID instead of a 5-digit code, the associated record is a generic alert. Generic alerts do not have unique prestored panels in the hardware monitor. See "Using NMVT Support for User-Written Programming" on page 60 for more information on generic alerts.

4. Determine which panel contains the text you want to change, as follows:
   - For a Recommended Action panel, the panel name (or panel alias) is BNI*xxxyy*, where *xxx* and *yy* are the codes you identified in step 4.
   - For an Event Detail panel, the panel name (or panel alias) is BNK*xxxyy*, where *xxx* and *yy* are the codes you identified in step 4.
5. Now determine whether BNI*xxxyy* or BNK*xxxyy* is an actual or alias panel name as follows:
   - **For MVS:**
     - Use an editor such as ISPF/PDF to examine the directory listing of panel names. This listing is in the IBM-supplied partitioned data set (PDS) named SYS1.BNJPNL1. The word *alias* appears to the right of those panel names that are aliases.
   - **For VM:**
     - Check the VM file list for the panel name. The filename is the same as the panel name, and the filetype is NCCFLST. If the panel name is an alias, it will not appear in the file list.
     - Use an editor such as XEDIT to examine the CSECT BNJBLKID file type ASSEMBLE. This lists the block IDS whose Recommended Action panels and Event Detail panels have at least one alias. (A sample BNJBLKID table is shown in Figure 15 on page 47.) If block ID *xxx* is on the list, check CSECT BNJAL*xxx* filetype ASSEMBLE, which lists all aliases for block ID *xxx*, to determine whether BNI*xxxyy* or BNK*xxxyy* is an alias for some other panel name. (A sample BNJAL*xxx* table is shown in Figure 16 on page 48.)
6. Go to the appropriate section, "Changing Panel Text" on page 48, "Changing From Alias to Actual" on page 49, "Deleting an Actual or Alias" on page 49, or "Adding an Actual or Alias" on page 50.

```
TITLE 'BNJBLKID:  LIST OF ALIAS TABLES BY BLOCK ID'
BNJBLKID CSECT
        EJECT
          DS  0F
NUMENT    DC  AL4((TABEND-TABSTART)/LENG)  NO. OF ENTRIES
TABSTART  EQU *
          DC  CL3'FED'
          DC  CL3'FEE'
          DC  CL3'FEF'
          DC  CL3'FE1'
          DC  CL3'FE2'
          DC  CL3'FE3'
          DC  CL3'FE4'
          DC  CL3'FFD'
          DC  CL3'FFE'
          DC  CL3'FFF'
          DC  CL3'FF2'
          DC  CL3'FF5'
          DC  CL3'FF6'
          DC  CL3'FF7'
          DC  CL3'FF8'
          DC  CL3'FF9'
          DC  CL3'005'
          DC  CL3'016'
          DC  CL3'017'
          DC  CL3'02D'
          DC  CL3'021'
          DC  CL3'022'
          DC  CL3'023'
          DC  CL3'03E'
          DC  CL3'036'
          DC  CL3'04A'
          DC  CL3'04B'
          DC  CL3'04C'
          DC  CL3'04D'
          DC  CL3'04E'
          DC  CL3'04F'
          DC  CL3'043'
          DC  CL3'044'
          DC  CL3'047'
          DC  CL3'049'
TABEND    EQU *
LENG      EQU 3                   ENTRY BYTE LENGTH
          END BNJBLKID
```

Figure 15. Sample BNJBLKID Table

```
         TITLE 'BNJAL036: ALIAS TABLE FOR BLOCKID 036'
BNJAL036   CSECT
          EJECT
           DS 0F
NUMENT     DC  AL4((TABEND-TABSTART)/LENG)    NO. OF PAIRS
*               REAL NAME       ALIAS NAME
TABSTART   EQU *
           DC  CL8'BNI03609',CL8'BNI0366D'
           DC  CL8'BNI03608',CL8'BNI0366C'
           DC  CL8'BNI03607',CL8'BNI0366B'
           DC  CL8'BNI03606',CL8'BNI0366A'
           DC  CL8'BNI03605',CL8'BNI03669'
           DC  CL8'BNI03605',CL8'BNI03671'
           DC  CL8'BNI03605',CL8'BNI0360D'
           DC  CL8'BNI03604',CL8'BNI03668'
           DC  CL8'BNI03604',CL8'BNI03670'
           DC  CL8'BNI03604',CL8'BNI0360C'
           DC  CL8'BNI03603',CL8'BNI03667'
           DC  CL8'BNI03602',CL8'BNI03666'
           DC  CL8'BNI03601',CL8'BNI03665'
           DC  CL8'BNI0360B',CL8'BNI0366F'
           DC  CL8'BNI0360A',CL8'BNI0366E'
           DC  CL8'BNK03609',CL8'BNK0366D'
           DC  CL8'BNK03608',CL8'BNK0366C'
           DC  CL8'BNK03607',CL8'BNK0366B'
           DC  CL8'BNK03606',CL8'BNK0366A'
           DC  CL8'BNK03605',CL8'BNK03669'
           DC  CL8'BNK03604',CL8'BNK03668'
           DC  CL8'BNK03603',CL8'BNK03667'
           DC  CL8'BNK03602',CL8'BNK03666'
           DC  CL8'BNK03601',CL8'BNK03665'
           DC  CL8'BNK0360D',CL8'BNK03671'
           DC  CL8'BNK0360C',CL8'BNK03670'
           DC  CL8'BNK0360B',CL8'BNK0366F'
           DC  CL8'BNK0360A',CL8'BNK0366E'
TABEND     EQU *
LENG       EQU 16                  ENTRY PAIR BYTE LENGTH
           END BNJAL036
```

Figure 16. Sample BNJALxxx Table

Changing Panel Text

If BNI*xxxyy* or BNK*xxxyy* is an actual panel name (not an alias), follow these steps to change the panel wording.

- **For MVS:**

  1. Use an editor such as ISPF/PDF to edit the PDS member containing the panel. The PDS name is SYS1.BNJPNL1 (unless it was changed during installation), and the member name is the same as the panel name.

  2. Save the changed member.

1 Do not alter the number of noncomment lines. BNI*xxxyy* panels must contain exactly 14 noncomment lines; BNK*xxxyy* panels must contain exactly 7 noncomment lines. Comment lines contain an asterisk (*) in column 1.

- **For VM**:
  1. Use an editor such as XEDIT to edit the CMS file containing the panel. The filename is the same as the panel name and the filetype is NCCFLST.
  2. Save the changed file.

The changes you have made will now apply to *all* event conditions that use the panel or any of its aliases.

Changing From Alias to Actual

If you want to make a panel that now appears under an alias name into an actual panel, follow these steps:

Note: For more information on MVS utilities, see *MVS/Extended Architecture Data Administration: Utilities*. For more information on JCL see *MVS/Extended Architecture Job Control Language User's Guide.*

- **For MVS**:
  1. Use an editor such as ISPF/PDF to edit the PDS member containing the panel alias. The PDS name is SYS1.BNJPNL1 (unless it was changed during installation), and the alias member name is the same as the panel name.
  2. Save the changed member. TSO will convert the panel alias into an actual panel.
- **For VM**:
  1. Find out if BNI*xxxyy* or BNK*xxxyy* is the only alias for block ID *xxx*. If it is:
     a. Erase the CSECT table BNJAL*xxx* filetype ASSEMBLE.
     b. Delete the block ID entry from the table BNJBLKID filetype ASSEMBLE.
     c. Assemble and relink BNJBLKID.

     If BNI*xxxyy* or BNK*xxxyy* is *not* the only alias for block ID *xxx*:

     a. Erase the alias entry from CSECT table BNJAL*xxx*.
     b. Assemble and relink BNJAL*xxx*.
  2. Create a CMS file with filename BNI*xxxyy* or BNK*xxxyy* and filetype NCCFLST, by copying the file from the actual panel for which BNI*xxxyy* or BNK*xxxyy* was an alias.
  3. Use XEDIT to make the desired changes to the new file.
  4. Save the changed file.

A new actual panel has now been created under the name that was formerly the alias name.

Deleting an Actual or Alias

To delete an actual or alias panel name:

- **For MVS**, either:
  - Delete the PDS member containing the actual or alias panel name. The PDS name is SYS1.BNJPNL1 (unless it was changed during installation), and the member name is the same as the panel name.
  - Use the utility IEHPROGM. For example, to delete aliases BNK04B2E and BNK04B2F using this utility, you might code the following:

```
//DELMEBR2 JOB  MSGLEVEL=(1,1)
//STEP1    EXEC  PGM=IEHPROGM
//SYSPRINT  DD  SYSOUT=A
//DS1   DD  VOL=SER=vsnum,DISP=SHR,UNIT=device_type
//SYSIN DD *
 SCRATCH VOL=device_type=vsnum,DSNAME=panel_dsname,
        MEMBER=BNK04B2E
//STEP2    EXEC  PGM=IEHPROGM
//SYSPRINT  DD  SYSOUT=A
//DS1   DD  VOL=SER=vsnum,DISP=SHR,UNIT=device_type
//SYSIN DD *
 SCRATCH VOL=device_type=vsnum,DSNAME=panel_dsname,
        MEMBER=BNK04B2F
/*
```

In this example, *device_type* is the device type, *vsnum* is the volume serial number on which the data set resides, and *panel_dsname* is the name of the data set containing the panels. For more information on MVS utilities, see *MVS/Extended Architecture Data Administration: Utilities*. For more information on JCL, see *MVS/Extended Architecture Job Control Language User's Guide*.

- **For VM**:
  - To delete an actual panel name, erase the CMS file containing the panel you want to delete. The filename is BNI*xxxyy* or BNK*xxxyy*, and the filetype is NCCFLST.
  - To delete an alias panel name:
    - If BNI*xxxyy* or BNK*xxxyy* is the only alias for block ID *xxx*:
      1. Erase the CSECT table BNJAL*xxx* filetype ASSEMBLE.
      2. Delete the block ID entry from table BNJBLKID filetype ASSEMBLE.
      3. Assemble and relink BNJBLKID.
    - If BNI*xxxyy* or BNK*xxxyy* is *not* the only alias for block ID *xxx*:
      1. Delete the alias name from table BNJAL*xxx*.
      2. Assemble and relink BNJAL*xxx*.

Adding an Actual or Alias

If you want BNI*xxxyy* or BNK*xxxyy* to be a new (or replacement) panel name or alias, follow these steps:

Note: For more information on MVS utilities, see *MVS/Extended Architecture Data Administration: Utilities*. For more information on JCL see *MVS/Extended Architecture Job Control Language User's Guide*.

- **For MVS**, either:
  - Add an actual panel (and name) (*not* an alias), using an editor such as ISPF/PDF to copy an existing panel that is similar to the desired one. Then make the required changes to the new panel.
  - Add either an actual panel or an alias, using the utility IEBUPDTE. For example, to add BNK04B2E as an alias of BNK04B2A using this utility, you might code the following:

```
//PANELS JOB MSGLEVEL=1,MSGCLASS=A
//UPDATE1 EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD SYSOUT=A
//SYSUT2 DD DSN=panel_dsname,DISP=SHR,UNIT=device_type,
//       VOL=SER=vsnum
//SYSIN DD *
./ ADD NAME=BNK04B2A
 DETAIL DESCRIPTION: THE ERROR ANALYSIS MICROCODE
 HAS DETECTED AN INVALID ERROR LOG ENTRY.



    LOG ENTRY 0-3      4-7               8-11
****************************************************
                              .
                              .
                              .
*
*
*
***************LAST LINE OF PDS MEMBER************
./  ALIAS  NAME=BNK04B2E
/*
```

In this sample, *panel_dsname* is the name of the data set where the panel is stored, and *vsnum* is the volume serial number on which the data set resides. Although the sample defines only one new alias, up to 15 are allowed. For more information on MVS utilities, see *MVS/Extended Architecture Data Administration: Utilities*. For more information on JCL, see *MVS/Extended Architecture Job Control Language User's Guide*.

- **For VM**:
  - To add a panel (and name), create a CMS file with filename BNI*xxxyy* or BNK*xxxyy*, and filetype NCCFLST. Insert the comments and noncomment text that you want. Alternatively, you can copy an existing file that is similar to the desired one and make the required changes to the new file.
  - To add a panel alias:
    - If block ID xxx exists in CSECT BNKBLKID filetype ASSEMBLE:
      1. Add an alias entry for BNI*xxxyy* or BNK*xxxyy* in CSECT BNJAL*xxx* filetype ASSEMBLE, following the format in the sample table, BNJAL036, shown in Figure 16 on page 48.
      2. Assemble and relink CSECT BNJAL*xxx*.
    - If block ID *xxx* does not exist in CSECT BNKBLKID:
      1. Add block ID *xxx* to CSECT BNJBLKID.
      2. Assemble and relink CSECT BNJBLKID.
      3. Create CSECT BNJAL*xxx* with an entry for alias BNI*xxxyy* or BNK*xxxyy*, following the format in the sample table, BNJAL036, shown in Figure 16 on page 48.
      4. Assemble and relink CSECT BNJAL*xxx*.

Alert Messages

To change the EVENT DESCRIPTION:PROBABLE CAUSE text of any selection on an Alerts-Static, Alerts-History, Alerts-Dynamic, Event Detail, or Most Recent Events panel that is not associated with generic alerts, follow these steps:

1. For the text you want to change, determine its associated event and identify a resource that the event has been logged against.
2. For the resource you identified in step 1, display the Alerts-Static, Alerts-History, Alerts-Dynamic, Event Detail, or Most Recent Events panel.
3. Enter sel# C, where *sel#* is the selection number of the event associated with the text you want to change.
4. Examine the 5-digit code, *xxxyy*, that the NetView program returns.

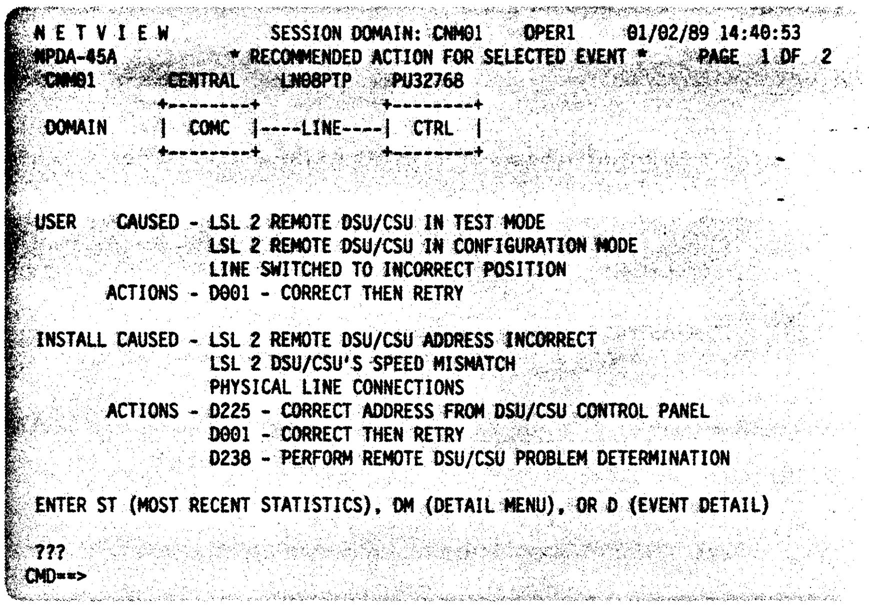
xxx	is the NetView-designated product code, or block ID, for the resource. (Block ID s are listed in *NetView Problem Determination and Diagnosis.*)
yy	is an individual panel identifier.

   Note: If you get a PRODUCT ID and ALERT ID instead of a 5-digit code, the associated record is a generic alert. Generic alerts do not have unique prestored EVENT DESCRIPTION:PROBABLE CAUSE text messages in the hardware monitor. See "Using NMVT Support for User-Written Programming" on page 60 for more information on generic alerts.
5. Use an editor such as ISPF/PDF (for MVS) or XEDIT (for VM) to retrieve and edit the CSECT that contains the text you want to change. The name of the CSECT is BNJVM*xxx* (for MVS, PDS member in SYS1.BNJSRC1, for VM, filetype ASSEMBLE) where *xxx* is the block ID you identified in step 4.
6. Locate the message text within BNJVM*xxx*. The message number for this text is the decimal equivalent of *yy*, where *yy* is the hexadecimal identifier you determined in step 4.
7. Change the assembler language macro DSIMDS for the text you want to change. For the syntax of DSIMDS, see *NetView Customization: Using Assembler.*
8. Save the changed CSECT.
9. Reassemble the CSECT, and link-edit it into the load module of the same name.

Overlaying Recommended Action Numbers

Recommended Action numbers (E*nnn* and I*nnn*) displayed from a generic alert can be overlayed with a different number. This section tells how to overlay default generic alert I-numbers and E-numbers on hardware monitor Recommended Action panels with sending product-unique action numbers.

On Recommended Action panels of the hardware monitor, each recommended action is identified with a special action number. Figure 17 on page 53 shows a sample Recommended Action panel with three recommended actions (D225, D001, and D238).



```
N E T V I E W          SESSION DOMAIN: CNM01    OPER1     01/02/89 14:40:53
NPDA-45A           * RECOMMENDED ACTION FOR SELECTED EVENT *      PAGE  1 OF  2
CNM01      CENTRAL    LN08PTP    PU32768
            +--------+            +--------+
 DOMAIN     | COMC   |----LINE----|  CTRL  |
            +--------+            +--------+


USER    CAUSED - LSL 2 REMOTE DSU/CSU IN TEST MODE
                 LSL 2 REMOTE DSU/CSU IN CONFIGURATION MODE
                 LINE SWITCHED TO INCORRECT POSITION
       ACTIONS - D001 - CORRECT THEN RETRY

INSTALL CAUSED - LSL 2 REMOTE DSU/CSU ADDRESS INCORRECT
                 LSL 2 DSU/CSU'S SPEED MISMATCH
                 PHYSICAL LINE CONNECTIONS
       ACTIONS - D225 - CORRECT ADDRESS FROM DSU/CSU CONTROL PANEL
                 D001 - CORRECT THEN RETRY
                 D238 - PERFORM REMOTE DSU/CSU PROBLEM DETERMINATION

ENTER ST (MOST RECENT STATISTICS), DM (DETAIL MENU), OR D (EVENT DETAIL)

???
CMD==>
```

Figure 17. Recommended Action Panel for Selected Event

The ACTION command list may be used to get more information on a recommended action displayed in the hardware monitor. See Chapter 3, "Modifying and Creating Online Panel Interfaces" on page 35 for information on how to modify the help panels displayed by use of the ACTION command list. D*nnn*, E*nnn*, and I*nnn* are recommended action numbers found on the Recommended Action panels. The following describes what would be displayed by the ACTION command list for the different recommended action numbers:

ACTION Dnnn	Displays an IBM-supplied, detailed description of a recommended action.
ACTION Ennn	Displays a detailed description of a recommended action created by your system programmer for a user-defined generic alert action.
ACTION Innn	Displays a detailed description of a recommended action created by your system programmer for an IBM-defined generic alert action.

Because details of a particular generic alert Recommended Action may vary depending on the sending product, action number panels cannot be provided for all possible generic actions. Therefore, on NetView Recommended Action panels built for generic alerts, each recommended action is preceded by an I-number (IBM-supplied action) or an E-number (user-supplied action).

I-number and E-number actions do not have NetView-supplied panels associated with them. However, the NetView program allows users to overlay I/E-numbers with action numbers to create panels that are specific to the sending product. This is done by modifying table BNJDNUMB in SYS1.BNJPNL2 (for MVS) or BNJDNUMB NCCFLST * (for VM) and creating BNJ*wwwwww* PDS members (MVS) or BNJ*wwwwww* NCCFLST * files (VM) (the * denotes filemode). In the rest of this section, BNJDNUMB and BNJ*wwwwww* refer to a PDS member for MVS and to a CMS file, filetype NCCFLST, for VM.

User Interfaces

BNJDNUMB

BNJDNUMB correlates a product-set identification (PSID) with a unique file or PDS member (BNJ*wwwww*) that contains the action numbers to use for this product. To modify BNJDNUMB, use an editor such as ISPF/PDF (for MVS) or XEDIT (for VM).

Note: If the NetView program receives a generic alert whose PSID does not exist in BNJDNUMB, the default I-number or E-number is not modified.

The format for BNJDNUMB is:

```
xxx
yyyyyyyyy BNJwwwww     comment
yyyyyyyyy BNJwwwww     comment
```

The symbol definitions are:

xxx	The number of entries in BNJDNUMB. This entry must begin in column 1 (three characters with leading zeros).
yyyyyyyyy	Up to nine characters representing the PSID. This entry must begin in column 1.
BNJ*wwwww*	The name of the PDS member (MVS) or file (VM), beginning in column 11, that contains generic alert recommended action code points and associated action numbers. Names such as BNJDNUM2, BNJDNUM3, and so forth, are recommended; however, any unique name can be used. BNJDNUM1 is already used for generic alerts produced by the hardware monitor.

Entries in BNJDNUMB must be in ascending order. Comment lines contain an asterisk (*) in column 1. The first line cannot be a comment line. Figure 18 shows a sample of BNJDNUMB provided by the NetView program. The 7-character NetView PSIDs for the various operating systems all map to BNJDNUM1, the action number file or PDS member used by the NetView program.

```
003
5664204   BNJDNUM1     NETVIEW R3 VM
5665361   BNJDNUM1     NETVIEW R2 MVS/370
5665362   BNJDNUM1     NETVIEW R3 MVS/XA
```

Figure 18. Sample BNJDNUMB

Determining the PSID: Because the sending product can be either a hardware product or a software product, the PSID is defined as follows:

- For hardware products, the PSID is defined with the four numeric characters identifying the machine type found in the X'00' subfield, Hardware Product Identifier (located in the first X'11' subvector of the first X'10' subvector in the generic alert).
- For software products, the PSID is defined with the nine uppercase alphameric characters of the serviceable component identifier in the X'02' subfield, Software Product Serviceable Component Identifier (located in the first X'11' subvector of the first X'10' subvector in the generic alert).

  Note: If the X'02' subvector does not exist, use the seven uppercase alphameric characters of the program product number in the X'08' subvector, Software Product Program Number (located in the first X'11' subvector of the first X'10' subvector in the generic alert).

Two methods are available to determine the PSID of a generic alert that has been logged to the hardware monitor data base.

1. Select sel# C from Alerts Static, Alerts History, or Most Recent Events panels to display a message containing the PSID.
2. The sending PSID is also displayed on page 1 of the PSID panel. This panel can be reached by a selection from the Event Detail Menu.

BNJwwwww

Each BNJ*wwwww* contains generic alert recommended action code points and associated action numbers. To create the BNJ*wwwww* files/members specified in table BNJDNUMB, use an editor such as ISPF/PDF (for MVS) or XEDIT (for VM). For MVS, each BNJ*wwwww* PDS member should be stored in the first data set in the concatenation string for the DD statement BNJPNL2. This DD statement is in the NetView startup procedure. For VM, each BNJ*wwwww* file should be stored on a minidisk accessed by the NetView virtual machine.

For MVS, if you want to modify or create a panel while the NetView program is running, define your panel data set without secondary extents. Otherwise, a panel might be filed in a new extent, and you would have to close and restart the NetView program to use this panel.

Following is the format for BNJ*wwwww*.

```
xxxx      yyyyyyyy  dnum
xxxx      yyyyyyyy  dnum
xxxx      yyyyyyyy  dnum
```

The symbols are defined as follows:

xxxx	The 4-character generic alert recommended action code point (the EBCDIC version of the recommended action code point as defined by the generic alert architecture). This field must begin in column 1.
yyyyyyyy	The 8-character alert ID number (EBCDIC version of the alert ID number as defined in the X'92' subvector architecture). This field is optional, but, if present, must begin in column 11.
dnum	The 4-character unique action number. This field begins in column 21. Action numbers can be any combination of four EBCDIC characters. The limiting factor of the action number is the ability of the ACTION command list to use these four characters and to display the associated panel.

Entries in each BNJ*wwwww* file/member must be in ascending hexadecimal order. If an invalid (nonhexadecimal) number is used, it will be skipped over.

The BNJ*wwwww* file/member specified in BNJDNUMB for the alert sender will be searched serially until a match is found or the end of the file is reached. Once the first * is found in column 1, the serial searching stops.

Blanks could be placed in the alert ID field, along with specific alert IDs, for a particular action code point. Figure 19 shows a sample BNJ*wwwww* user-defined table.

```
1002                D562
1002      93987791  D890
1002      D2556B79  D777
```

Figure 19. Sample BNJwwwww User-Defined Table

For alert D2556B79, the code point 1002 uses D777 as its action number. For alert 93987791, code point 1002 uses D890 as its action number. For all other alerts from this sending product, code point 1002 uses D562 as its action number.

Color and Highlighting

For the hardware monitor displays, you can alter the color, highlighting, and intensity of the display's text. You can also cause the display to produce an audible alarm. Consider the needs of the display users, though, before you modify these four attributes as assigned by the NetView program.

Note: You should not change any attribute's length, row placement, or column placement, since the results are unpredictable.

For any string of display text that is preceded by a blank, you can modify up to four attributes:

Attribute	**Meaning**
Color	Text appears red, yellow, blue, white, green, turquoise, or pink.
Highlighting	Text is shown underscored, blinking, or in reverse video.
Intensity	Text appears more intense (on monochrome terminals only).
Alarm	Text causes an audible alarm at the user's terminal.

You can change these attributes for specific displays or for all displays. For example, you could select a single color for prompt lines on all displays.

The procedure for modifying these attributes begins with a color map. A color map is a table that causes characters representing the various attributes to be imbedded in a color buffer. These characters in the color buffer control the appearance of the display.

Selecting the Color Map

The first step in modifying a hardware monitor display is to determine which color map controls the display you want to change. Appendix A, "Color Maps for Hardware Monitor Panels" on page 79 contains a matrix of the display name, display number, and color map for hardware monitor displays.

After you have identified the color map you need, edit it using an editor such as ISPF (for MVS) or XEDIT (for VM). For MVS, the color maps are contained in the PDS named SYS1.BNJPNL2 (unless the name was changed during installation); the member name is the color map name. For VM, the file name is the color map name and the file type is NCCFLST.

Note: If you want a particular attribute to apply to the same portion of *all* displays, create color map BNJOVERW, which overwrites all other display-specific color maps. Be sure to test the results of BNJOVERW on all displays before putting it into your production system. This map may produce unexpected results.

Modifying the Color Map

After selecting the color map, you can modify it. A color map consists of a series of lines of data, called map elements. The top line of a color map is always the number of subsequent map elements. Map elements begin in column 1 and are paired with comments that begin in column 41.

Each map element specifies, for a particular display row, the attribute, the attribute's placement in the row, and the length in characters. Each item in the map is followed by a comma except for the last one, which is followed by a period.

Note: You should not change any attribute's length, row placement, or column placement, since the results are unpredictable.

Figure 20 shows a sample color map. The numerical references are explained on pages 57 – 59.

```
13.[1]                                  NUMBER OF ELEMENTS IN TABLE
1,1,1,79,BLU,[2]                        NETVIEW HEADER
1,2,1,14,BLU,                           SCRN ID
2,2,16,64,HIG,WHI,                      SCRN TITLE
1,3,1,7,BLU,                            DOMAIN
1,3,9,71,TUR,
1,5,1,79,BLU,                           HEADING
99,SIZE-0-7,2,[3]                       REPETITION
2,6,1,4,HIG,WHI,                        SEL #
1,6,6,74,TUR,                           DATA
1,SIZE-4,1,50,BLU,[4]                   PROMPT LINE
2,SIZE-4,52,1,HIG,WHI,                  PROMPT LINE
1,SIZE-4,54,26,BLU,                     PROMPT LINE
1,SIZE-3,1,79,BLU.                      PROMPT LINE
```

Figure 20. Sample Color Map

The components of a color map, illustrated in Figure 20, are as follows:

[1] The first item in the color map must be the number of subsequent lines of data, or map elements, in the map. A map can have any number of map elements. The sample map has 13 map elements.

[2], [3], and [4] illustrate the three types of map elements, which are described below:

[2] This type of map element contains attribute information in the following format:

- The first item is the number of attributes in the map element. This number can be from one to four. A map element can have just one attribute such as pink color or any combination, such as pink color and underscoring. The sample map element has one attribute, blue color (BLU).
- The second item is the number of the display row that is to reflect the attribute. In the sample, the attribute is to appear in row 1.
- The third item is the number of the display column in which the attribute character is to be placed. In the sample, the attribute character is to be placed in column 1. Consequently, the displayed text will begin in column 2.

  Note: Be sure that the display text you want to modify has a blank space immediately to the left of it. Otherwise, the character representing the attribute in the color buffer will overwrite some of the display text, and some characters will appear as blanks instead. For example, you cannot make the colon a different color from the text in this string:

```
EVENT DESCRIPTION:PROBABLE CAUSE
```

- The fourth item is the maximum character length of the attribute. In the sample, the specified attribute covers 79 characters on the display, or columns 2 through 80.
- The last item is the attribute or sequence of several attributes. In the sample, the color blue is the specified attribute. You can specify up to four attributes, but only one from each category. If you want multiple attributes to apply to the same character or string, you must specify the attributes in the order shown below for each category:

  1. Alarm:
     - ALM produces an audible alarm
  2. Intensity:
     - HIG intensifies the color
     - NOH returns the color to normal intensity
  3. Highlighting:
     - REV shows the character or string in reverse video
     - UND underlines the character or string
     - BLI causes the character or string to blink
  4. Color:
     - RED produces red
     - YEL produces yellow
     - BLU produces blue
     - WHI produces white
     - GRE produces green
     - TUR produces turquoise
     - PIN produces pink.

To summarize, this map element makes the text in row 1, columns 2 through 80, blue. As the map element's corresponding comment confirms, this blue string of text is the display header.

3 This type of map element uses the repetition factor option to copy the attribute(s) specified for a particular row onto subsequent rows. A repetition map element uses the following format:

- The number 99 signals the repetition of an element.
- In SIZE-x-y,
  - SIZE represents the total number of rows in the display. Use the word SIZE as shown; do not replace it with a number.
  - *x* is the number of unused or blank lines between the end of display data and the prompt line. In the sample, no blank or unused lines occur between the end of display data and the prompt line.
  - *y* is the number of the starting row that is to copy, or repeat, the attribute(s) from the row above it. In the sample, attributes from row 6 are to be repeated on the subsequent rows, starting with row 7.
- The last item is the number of attributes on row 6 to be repeated. In the sample, the two attributes specified in the map for row 6 are to be repeated.

To summarize, this map element copies the two attributes specified for row 6 onto subsequent rows, starting at row 7 and continuing to the prompt line.

4 This last type of map element uses the variable row placement option to specify the row that is to contain the attribute. This option uses the following format:

- The first item is the number of attributes in the map element. This number can be from 1 to 4. In the sample, the map element has one attribute, blue color (BLU).
- The second item, SIZE-x, indicates the display row that is to reflect the attribute, where:
  - SIZE represents the total number of rows in the display. Use the word SIZE as shown; do not replace it with a number.
  - *x* is the number of lines above the command line.

  For example, for the Alerts-Static display:

  - SIZE-4 is the first prompt line
  - SIZE-3 is the second prompt line
  - SIZE-2 is the message line
  - SIZE-1 is the NetView status line
  - SIZE-0 is the command line.

    Note: So that no bytes are overwritten, be sure that the command line is defined on byte 80 of the NetView status line.

  In the sample, the attribute is to appear on the first prompt line.
- The third item is the number of the display column in which the attribute character is to be placed. In the sample, the attribute character is placed in column 1. Consequently, the displayed text begins in column 2.

  Note: Be sure that the display text you want to modify has a blank space immediately in front of it. Otherwise, the character representing the attribute in the color buffer will overwrite some of the display text, and some characters will appear as blanks instead.
- The fourth item is the maximum character length of the attribute. In the sample, the specified attribute covers 50 characters on the display.
- The last item is the attribute or sequence of several attributes. You may specify up to four attributes, but only one from each category. If you want multiple attributes to apply to the same character or string, you must specify the attributes in the order shown on page 58. In the sample, color blue is the specified attribute.

To summarize, this map element makes the text in the first prompt line, columns 2 to 51, blue.

Prompt Highlight Tokens

The prompt highlight token table, BNJPROMP, is located in the PDS named SYS1.BNJPNL2 for MVS and in BNJPROMP filetype NCCFLST for VM, with the color maps. It can also be modified. The maximum size of the table is 25 prompts, with the prompt being a 15-byte character field. If you decide to modify the table, use the comment column for comments about the table. Color is a 3-byte character field beginning at column 20. You can select only those colors that are valid in the color maps. The table has the following format:

Table 8. Prompt Highlight Tokens

Prompt Token	Color	Comment
SEL#	WHI	PROMPT SEL#
LDM	WHI	PROMPT LDM
LSL1	WHI	PROMPT LSL1
LSL2	WHI	PROMPT LSL2
RESNAME	WHI	PROMPT RESNAME
RESNAME1	WHI	PROMPT RESNAME1
RESNAME2	WHI	PROMPT RESNAME2
'A'	WHI	PROMPT A
'B'	WHI	PROMPT B
'P'	WHI	PROMPT P
'EV'	WHI	PROMPT EV
'ST'	WHI	PROMPT ST
'DM'	WHI	PROMPT DM
'M'	WHI	PROMPT M
'DEL'	WHI	PROMPT DEL
'S'	WHI	PROMPT S
'D'	WHI	PROMPT D
'R'	WHI	PROMPT R

The table is read into storage at initialization. You can redefine the prompt highlight tokens or add new ones up to a maximum of 25. You will receive a message if the table was not successfully read at initialization.

Using NMVT Support for User-Written Programming

Network Management Vector Transport (NMVT) support enables user-written programming to report errors to the hardware monitor through generic alerts. In the past, Recommended Action panels, Event Detail panels, and alert messages were stored at the host in the NetView program. Each pregeneric alert had a unique set of panels and messages. With generic alerts, coded information is contained in the NMVT, and this information (generic alert code points) is used to dynamically build the hardware monitor panels. Pregeneric alerts are mainly used for migration purposes. New user-defined alerts should be created using generic alerts. For more information related to major vectors and subvectors of an NMVT, refer to *Systems Network Architecture Product Formats*.

This section contains a sample generic alert (Figure 21 on page 63) and the associated panels (Figure 22 on page 64, Figure 23 on page 65, Figure 24 on page 66, and Figure 25 on page 67) that are built by the hardware monitor. How each panel is built is also described.

Hardware monitor support for user-defined pregeneric alerts is included for migration purposes only. Any new alerts should be created in the generic alert format.

User-Defined Alerts—Pregeneric

Sixteen block IDs (X'F00' – X'F0F'), which are part of NMVT Major Vector 0000, are reserved for generating user-defined alerts.

The hardware monitor reserves USER0bb – USERFbb for use as the corresponding 7-character software identifier in the Software Product Program Number (X'08') subfield of the first Product Identifier (X'11') subvector of the NMVT.

The hardware monitor allows a 1-byte alert description code (X'xx') within the Basic Alert (X'91') subvector of the NMVT. This code lets you further qualify the alert. Put your alert description code in the second byte of the 2-byte Alert Description Code field; the hardware monitor ignores the first byte of that field.

NMVT-to-Panel ID Mapping: Using the block ID derived from the Software Product Program Number and the alert description code, the hardware monitor maps the NMVT to the following:

A 14-line panel, which appears on the hardware monitor's Recommended Action panel for the NMVT. The 14-line panel's PDS member name is in the range between BNIF00*xx* and BNIF0F*xx*, where F00 to F0F is the allowable range of block IDs, and *xx* is the hexadecimal value of the alert description code. The lines can be up to 80 characters long.

A 7-line panel, which appears on the hardware monitor's Event Detail panel for the NMVT. The 7-line panel's PDS member name is in the range between BNKF00*xx* and BNKF0F*xx*, where F00 to F0F is the allowable range of block IDs, and *xx* is the hexadecimal value of the alert description code.

The first eight translated characters of each of the first three X'A0' or X'A1' qualifier subvectors are displayed on an eighth line just below the Event Detail panel. Write the event detail messages with titles on the seventh line as needed to describe the qualifiers.

A 48-byte alert description, which appears on the Alerts-Dynamic, Alerts-Static, Alerts-History, Event Detail, and Most Recent Events panels. The 48-byte text descriptions for a block ID are in a NetView message CSECT whose link-edit load module name is in the range between BNJVMF00 and BNJVMF0F.

Panel Formats: For each new Recommended Action panel or Event Detail panel, use the same formats as in the existing panels when adding a panel to the NetView panel library or to a concatenated user library.

For each new 48-byte alert description CSECT, use the same format as an existing BNJVM*xxx* CSECT. The BNJVM*xxx* CSECT's are coded using the macro DSIMDS. No variable substitution is permitted for the 48-byte alert descriptions.

User-Defined Alerts—Generic

Generic alerts allow coded alert data to be transported within the alert, thus eliminating the need for stored panels. The coded data can be either:

- An index into predefined tables containing short units of text to be used in building a display, or
- Textual data to be displayed directly on the display.

In both cases, the data is completely independent of the NetView program. The text strings indexed by the code points and the display of textual data that was sent in the alert are in exactly the same format regardless of what product sent the alert. Also, the same terminology will be used to define similar problems within different products, since each product will typically use terminology which has been defined by IBM.

Generic alerts produce basically the same Alerts, Recommended Action, and Detail panels as the hardware monitor's previous alert support, but the panels are built dynamically rather than using stored panels. Code points index into the tables defined by IBM and the user.

The alert description and probable cause code points are used to build the hardware monitor Alerts-Dynamic, Alerts-Static, Alerts-History, Event Detail, and Most Recent Events panels. The user cause, install cause, failure cause, and recommended action code points are used to build the hardware monitor Recommended Action panel. The detail data code points are used to identify the qualifiers which may appear on either the hardware monitor Recommended Action or Event Detail panel. Products typically use the same set of architected product-independent terminology to define their Alert, Recommended Action, and Detail panels. Text data transported in the NMVT is displayed on the Event Detail panel.

The NetView program supports two sets of generic alert tables. One is a set of user tables into which you can add code points (see "Creating User-Defined Generic Code Points" on page 68). The other is a set of tables that contain IBM-supplied code points (see *NetView Problem Determination and Diagnosis* for a list).

The seven IBM-supplied generic alert tables are:

- BNJ92TBL—alert description code points
- BNJ93TBL—probable cause code points
- BNJ94TBL—user cause code points
- BNJ95TBL—install cause code points
- BNJ96TBL—failure cause code points
- BNJ81TBL—recommended action code points
- BNJ82TBL—detail data code points.

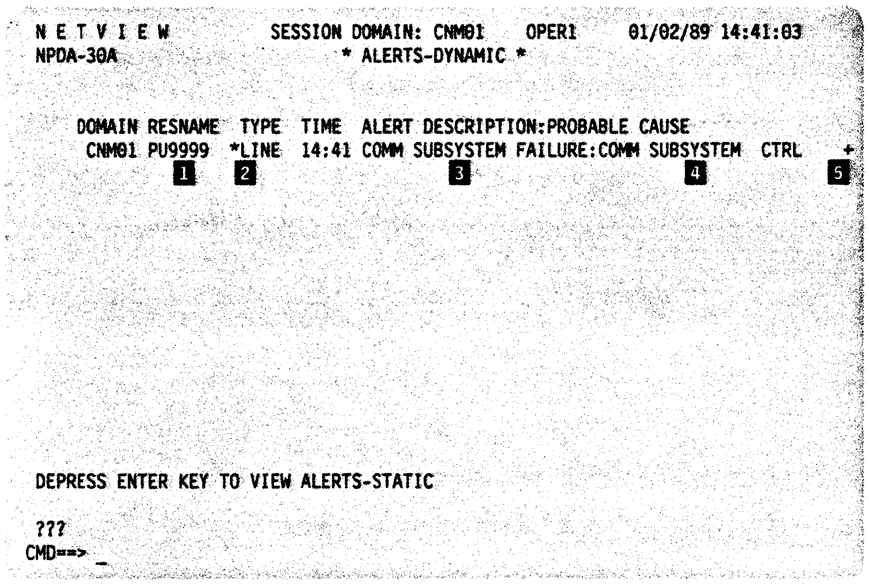
Using the GENALERT Command: You can use the GENALERT command to create your own alerts. The GENALERT command is described in *NetView Operation*.

Building Generic Alert Panels

Figure 21 is an example of a generic alert NMVT and the unique panels that are built using the information contained in the alert. Included after the example is a brief explanation of how each panel is built. For more information on NMVTs, see *Systems Network Architecture Network Product Formats.*

```
X'41038D5002000000'             Response Header
X'01230000'                     Major Vector Length and Key
X'0A0108105901020A2827'         01 SV - Date/Time
X'0B92000001'                   92 SV - Alert Description
X'1603'                           code point
X'1A2B3C4D'
X'0693'                         93 SV - Probable Cause(s)
X'0403'                           code point
X'2012'                           code point
X'1195'                         95 SV - Install Cause(s) and Action(s)
X'0601'                           01 SF - install cause(s)
X'1502'                             code point
X'13E1'                             code point
X'038391'                         83 SF - qualifier(s)
X'0681'                           81 SF - recommended action(s)
X'0101'                             code point
X'1504'                             code point
X'2796'                         96 SV - Failure Cause(s) and Action(s)
X'0601'                           01 SF - failure cause(s)
X'0503'                             code point
X'33C2'                             code point
X'068200'                         82 SF - qualifier(s)
X'61'                               code point
X'0004'
X'0C8200'                         82 SF - qualifier(s)
X'53'                               code point
X'11F0F0406040F1C6'
X'0A81'                           81 SF - recommended action(s)
X'0611'                             code point
X'0500'                             code point
X'3110'                             code point
X'00E1'                             code point
X'038321'                         83 SF
X'1705'                         05 SV - Resource Hierarchy
X'151000'                         10 SF
X'07D7E4F9F9F9F900F1'               name/type pair
X'07D3C9D5C5F0F440F9'               name/type pair
X'4D1000'                       10 SV - PSID
X'341104'                         11 SV - Product Identifier
X'0E02C1C3C661C9C2D44040F0F0F3'     02 SF - software product serviceable component ID
X'0804F0F1F0F2F0F3'                 04 SF - software product common level
X'0A06C1C3C661C9C2D440'             06 SF - software product common name
X'0A07C6C6C7C1C9E3D9F3'             07 SF - software product customization ID
X'07098603351225'                   09 SF - software product customization date and time
X'161101'                         11 SV - Product Identifier
X'130012'                           00 SF - hardware product identifier
X'F9F9F9F9F1F1C1F0F5'
X'F0C1F0C1F0C1F0'
X'1798'                         98 SV - Detailed Data
X'0782213400'                     82 SF - qualifier
X'0004'
X'0782000911'                     82 SF - qualifier
X'F2F2'
X'0782000E00'                     82 SF - qualifier
X'00DC'
X'2548'                         48 SV - Correlation
X'1060'                           60 SF - correlation for supporting data
X'D7C3C9C4D3E4F0F4'
X'05C3D5D4F0F1'
X'0D82'                           82 SF - qualifier
X'00DA11C3D6D4D460C5D9D9'
X'068200D1010F'                   82 SF - qualifier
X'3631'                         31 SV - Self Defining Text Message
X'060211340500'                 02 SF - Coded Character Set ID
X'0512C5D5E4'                   12 SF - National Language ID
X'032112'
X'2630'                         30 SF - Text Message
X'E3C8C9E240E2E4C2C6C9C5D3C440C9C4C5D5E3C9C6C9C5E240E3C8C540E3C5E7E340D4E2'
```

Figure 21. Sample Generic Alert Record



```
N E T V I E W          SESSION DOMAIN: CNM01    OPER1     01/02/89 14:41:03
NPDA-30A                          * ALERTS-DYNAMIC *

    DOMAIN RESNAME  TYPE  TIME  ALERT DESCRIPTION:PROBABLE CAUSE
     CNM01 PU9999  *LINE  14:41 COMM SUBSYSTEM FAILURE:COMM SUBSYSTEM  CTRL    +
                 1    2                       3                    4           5

DEPRESS ENTER KEY TO VIEW ALERTS-STATIC

???
CMD==> _
```

Figure 22. Sample of Alerts-Dynamic Display

An entry on the Alerts panels is built from a number of subvectors (X'92', X'93', and X'05').

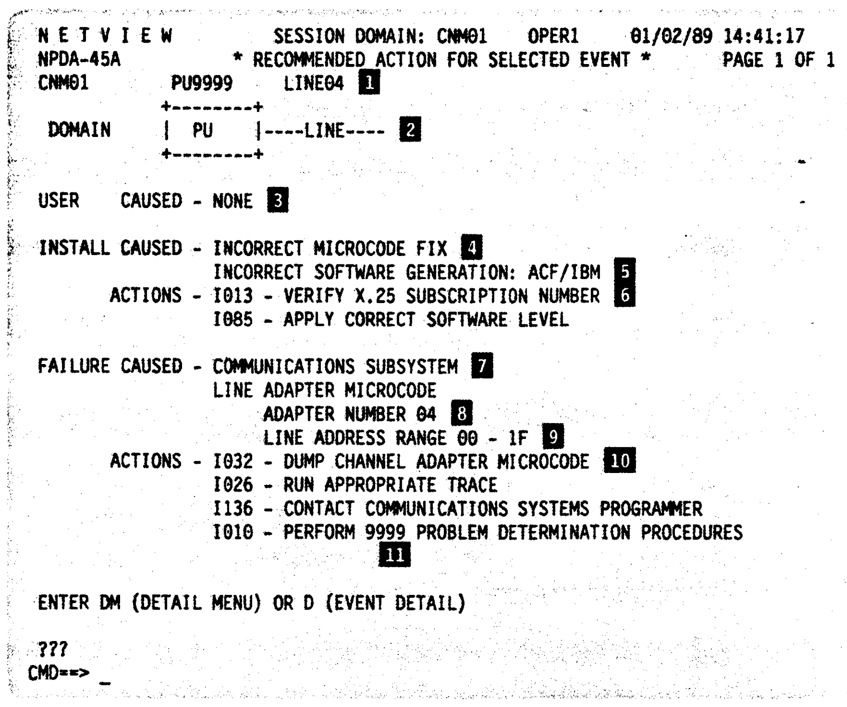
1 The RESNAME (PU9999) and TYPE (LINE) come from the last name/type pair in the X'05' subvector.

2 The * indicates that the RESNAME which precedes the type does not belong to the type. The type is always associated with the last name in the hierarchy, but the name depends on the manner in which the X'05' was coded. The Do Not Display Resource Name Indicator bit is set to 1 for the last name/type pair (subvector X'05', subfield X'10', second name/type pair, eighth byte, second bit). (See Figure 21 on page 63.) With the bit off, the previous resource name is displayed.

3 The ALERT DESCRIPTION (COMM SUBSYSTEM FAILURE) is derived from code point X'1603' in the X'92' subvector. The code point is used as an index into a table containing the alert description text messages.

4 The PROBABLE CAUSE (COMM SUBSYSTEM CTRL) is derived from code point X'0403' in the X'93' subvector. The code point is used as an index into a table containing the probable cause text messages.

5 The + is displayed because in this example the X'93' subvector contains more than one probable cause code point. The + indicates that more probable causes can be seen on the Event Detail panel.



```
N E T V I E W        SESSION DOMAIN: CNM01    OPER1     01/02/89 14:41:17
NPDA-45A          * RECOMMENDED ACTION FOR SELECTED EVENT *       PAGE 1 OF 1
CNM01        PU9999    LINE04 1
             +--------+
 DOMAIN      | PU     |----LINE---- 2
             +--------+

USER    CAUSED - NONE 3

INSTALL CAUSED - INCORRECT MICROCODE FIX 4
                 INCORRECT SOFTWARE GENERATION: ACF/IBM 5
       ACTIONS - I013 - VERIFY X.25 SUBSCRIPTION NUMBER 6
                 I085 - APPLY CORRECT SOFTWARE LEVEL

FAILURE CAUSED - COMMUNICATIONS SUBSYSTEM 7
                 LINE ADAPTER MICROCODE
                     ADAPTER NUMBER 04 8
                     LINE ADDRESS RANGE 00 - 1F 9
       ACTIONS - I032 - DUMP CHANNEL ADAPTER MICROCODE 10
                 I026 - RUN APPROPRIATE TRACE
                 I136 - CONTACT COMMUNICATIONS SYSTEMS PROGRAMMER
                 I010 - PERFORM 9999 PROBLEM DETERMINATION PROCEDURES
                                   11

ENTER DM (DETAIL MENU) OR D (EVENT DETAIL)

???
CMD==> _
```

Figure 23. Sample of Recommended Action for Selected Event

The Recommended Action panel is built from a number of subvectors (X'94', X'95', and X'96') and subfields (X'01', X'81', X'82', and X'83').

1 The resource names (PU9999 and LINE04) are taken directly from the X'05' hierarchy names list subvector. In this example, only names from the X'05' subvector were used because the Hierarchy Complete Indicator bit (byte 2 bit 0) in the X'05' subvector was set to 0. If this bit were set to 1, the NetView program would concatenate the names in the X'05' subvector to the names supplied by VTAM.

2 The resource types (PU and LINE) are derived by converting the type codes in the X'10' subfield of the X'05' subvector (X'F1' and X'F9') into displayable resource types. For more information on changing resource types, see "Adding or Modifying Resource Types" on page 71.

3 The X'94' subvector carries user-caused information. Because no X'94' subvector is in the sample NMVT, no user-caused information is displayed.

4 The two install-caused probable causes are built from code points (X'1502' and X'13E1') in the X'01' subfield within the X'95' subvector. Note that the E in the X'13E1' code point indicates an X'83' subfield is needed to complete the install cause.

5 The qualifier on the install cause (ACF/IBM) is displayed because of the X'83' subfield of the X'95' subvector. The X'83' subfield contains the value X'91' which indicates that the qualifier should be taken from the product ID subfield (X'06' Software Product Common Name) of the first product identifier subvector (X'11').

6 The two install-caused actions are taken from code points (X'0101' and X'1504') in the X'81' subfield of the X'95' subvector.

7 The two failure-caused probable causes are taken from code points (X'0503' and X'33C2') in the X'01' subfield of the X'96' subvector. Note that the C in the X'33C2' code point indicates that two X'82' subfields are needed to complete the failure cause.

8 ADAPTER NUMBER 04 is broken down from the first X'82' subfield in the X'96' subvector as follows:

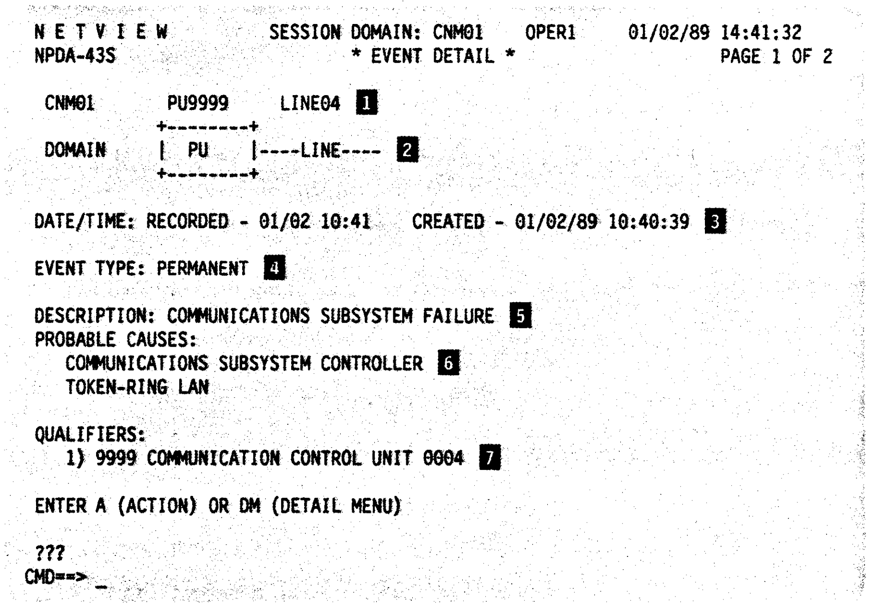
00	indicates no information will be taken from the PSID subvector
61	a code point for adapter number
00	indicates that hexadecimal data follows
04	hexadecimal data to be displayed.

9 LINE ADDRESS RANGE 00 - 1F is broken down from the second X'82' subfield in the X'96' subvector as follows:

00	indicates no information will be taken from the PSID subvector
53	a code point for line address range
11	indicates that EBCDIC data follows
F0F0406D40F1C6	EBCDIC data to be displayed.

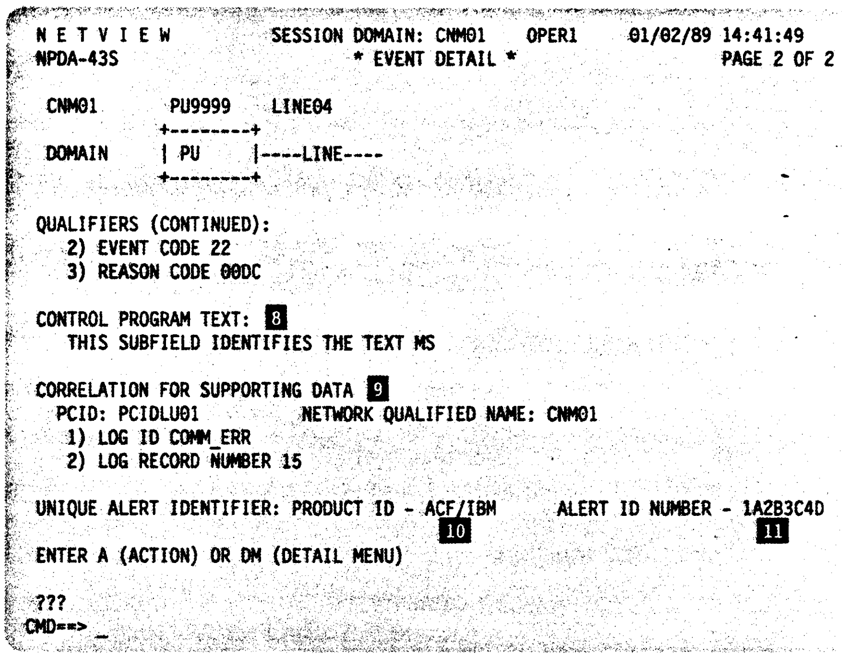
10 The failure-caused actions are taken from the code points (X'0611', X'0500', X'3110', and X'00E1') in the X'81' subfield of the X'96' subvector. Note that the E in the X'00E1' code point indicates that an X'83' subfield is needed to complete the failure cause.

11 The qualifier on the failure cause (9999) is displayed because of the X'83' subfield of the X'96' subvector. The X'83' subfield contains the value X'21' which indicates that the qualifier should be taken from the first hardware PSID subfield (X'00') of the PSID subvector (X'11').



```
N E T V I E W          SESSION DOMAIN: CNM01    OPER1     01/02/89 14:41:32
NPDA-43S                          * EVENT DETAIL *                 PAGE 1 OF 2

 CNM01       PU9999      LINE04 1
            +--------+
 DOMAIN     |  PU    |----LINE---- 2
            +--------+

DATE/TIME: RECORDED - 01/02 10:41    CREATED - 01/02/89 10:40:39 3

EVENT TYPE: PERMANENT 4

DESCRIPTION: COMMUNICATIONS SUBSYSTEM FAILURE 5
PROBABLE CAUSES:
   COMMUNICATIONS SUBSYSTEM CONTROLLER 6
   TOKEN-RING LAN

QUALIFIERS:
   1) 9999 COMMUNICATION CONTROL UNIT 0004 7

ENTER A (ACTION) OR DM (DETAIL MENU)

 ???
CMD==> _
```

Figure 24. Sample of Event Detail Display (page 1)



```
N E T V I E W          SESSION DOMAIN: CNM01    OPER1     01/02/89 14:41:49
NPDA-43S                         * EVENT DETAIL *                 PAGE 2 OF 2

 CNM01       PU9999    LINE04
            +--------+
 DOMAIN     | PU     |----LINE----
            +--------+

QUALIFIERS (CONTINUED):
   2) EVENT CODE 22
   3) REASON CODE 00DC

CONTROL PROGRAM TEXT: 8
   THIS SUBFIELD IDENTIFIES THE TEXT MS

CORRELATION FOR SUPPORTING DATA 9
  PCID: PCIDLU01           NETWORK QUALIFIED NAME: CNM01
   1) LOG ID COMM_ERR
   2) LOG RECORD NUMBER 15

UNIQUE ALERT IDENTIFIER: PRODUCT ID - ACF/IBM      ALERT ID NUMBER - 1A2B3C4D
                                        10                            11
ENTER A (ACTION) OR DM (DETAIL MENU)

???
CMD==> _
```

Figure 25. Sample of Event Detail Display (page 2)

The Event Detail panel is built from subvectors X'92', X'93', X'98', X'01', X'31', and X'48' and subfield X'82'.

1 The resource names (PU9999 and LINE04) are taken directly from the X'05' hierarchy names list subvector. In this example, only names from the X'05' subvector are used because the Hierarchy Complete Indicator bit (byte 2 bit 0) in the X'05' subvector was set to 0 (see Figure 21 on page 63). If this bit were set to 1, the NetView program would concatenate the names in the X'05' subvector to the names supplied by VTAM.

2 The resource types (PU and LINE) are derived by converting the type codes in the X'10' subfield of the X'05' subvector (X'F1' and X'F9') into displayable resource types. For more information on changing resource types, refer to "Adding or Modifying Resource Types" on page 71.

3 The DATE/TIME RECORDED is the time the record was logged to the hardware monitor data base. The field shows the time the record was created by the sending product. It is taken from the X'10' subfield of the X'01' subvector.

4 EVENT TYPE is derived from byte 4 (Alert Type) of the X'92' subvector.

5 DESCRIPTION is derived from the code point (X'1603') in the X'92' subvector as is the description on the Alerts panel. However, a longer version of the text is displayed on this screen.

6 PROBABLE CAUSES are taken from the code points (X'0403' and X'2012') in the X'93' subvector. Just as with the description, a longer version of the text is displayed than was displayed on the Alerts panel. Also, all of the probable causes are displayed.

7 QUALIFIERS are derived from the X'82' subfields of the X'98' subvector. Note that the NetView program ignores X'01' subfields and associated sub-subfields (including X'82') in a X'98' subvector. The qualifiers can be broken down as follows:

First X'82' subfield:

21	indicates the data should be taken from the first hardware PSID subfield (X'00') of the PSID subvector (X'11').
34	code point indicating communication control unit
00	indicates that hexadecimal data follows
0004	hexadecimal data to be displayed.

Second X'82' subfield

00	indicates no data will be taken from the PSID subvector
09	code point indicating event code
11	indicates that EBCDIC data follows
F2F2	EBCDIC data to be displayed.

Third X'82' subfield

00	indicates no data will be taken from the PSID subvector
0E	code point indicating reason code
00	indicates that hexadecimal data follows
00DC	hexadecimal data to be displayed.

Page 2 of the Event Detail panel contains the following information:

8 The control program text title is displayed because of subfield X'21' of subvector X'31'. The text itself is taken directly from subfield X'30' of the X'31' subvector and displayed on the screen.

9 The correlation for supporting data section is displayed from the X'48' subvector. Subfield X'60' specifies the network qualified procedure correlation identifier be used to uniquely identify a session. Two X'82' subfields identify the supporting data.

10 The product ID (ACF/IBM) is taken directly from the first product identifier (X'11') subvector in the first PSID (X'10') subvector. In this example, the Software Product Serviceable Component Identifier (X'02') subfield is used.

11 The alert ID number (1A2B3C4D) is taken from subvector X'92' bytes 7-10.

Creating User-Defined Generic Code Points

This section explains how to obtain problem determination support for devices and applications in your network using code points that the NetView program does not automatically support. You enter data into seven user tables that are shipped with the NetView program. The user tables code point range, X'E000' to X'EFFF', is reserved for your use; however, any code points not used in IBM-supplied tables can also be used. To create a help panel for a user-defined code point or an IBM-defined code point, follow the instructions under "Creating New Panels" on page 39.

Defining User Tables

For MVS, submit the NetView sample CNMSJM08 (found in the *Network Program Products Samples*), which allocates the PDS, CNM.CODE.POINTS, and is defined by USERLIB in CNMSJM07. CNMSJM08 contains seven members referred to as tables (see below).

For VM, running the NetView sample, CNMSVM08 (found in *Network Program Products Samples*), creates seven CMS files referred to as tables (see below). The filetype is NCCFLST. CNMSVM08 accepts the filemode as an input parameter. For example, coding `CNMSVM08 NCCFLST K` creates the seven tables on the minidisk accessed as K. The filemode default is A.

Table Formats

The seven user tables are:

- BNJ92UTB—alert description code points
- BNJ93UTB—probable cause code points
- BNJ94UTB—user cause code points
- BNJ95UTB—install cause code points
- BNJ96UTB—failure cause code points
- BNJ81UTB—recommended action code points
- BNJ82UTB—detail data code points.

Each table contains a different type of code point. The fourth and fifth characters of the table name identify the subvector or subfield that contains the code points.

An example of the data in the tables is shown in Figure 26. The numerical references are explained following the figure.

```
1    2
E0E2 x PROGRAM CHECK IN PROGRAM; 3
* this is a sample comment
4 0320 x PERSISTENT DCE CLEAR INDICATION DURING CALL  ESTABLISHMENT (T6 TIMER EXPIRED)
E0E1 Y MODEM $ POWERED OFF;
FFFF 5
```

Figure 26. Sample User-Defined Table

1 The code point is a 4-character hexadecimal number, starting in column 1. Valid characters are 0 through 9 and A through F. The code point range X'E000' to X'EFFF' has been reserved for your use. Unused code points outside this range can also be used, although they could be used by IBM in the future. **The last entry in the table must be code point** FFFF. Any entries after FFFF are not processed. One blank must follow the code point.

Note: Code points in table BNJ82UTB must be left-justified and padded with 0's. For example, you would enter code point 12 as 1200.

If the code point is not found in the IBM table, the equivalent user-defined table will be searched for the same code point. If a code point is received that is within the X'E000' to X'EFFF' range, *only* the user-defined table will be searched.

2 The text description appears in columns 8 through 72 on the hardware monitor displays. The maximum length varies as follows:

Probable cause	20 characters
Alert description	25 characters
Detail data	40 characters
Others	108 characters.

To continue the text on the next line, start in column 2.

If you want to imbed data from the X'82' or X'83' subfield in the text, mark the imbed position with a dollar sign ($). No variable substitution can be done for probable cause text (BNJ93UTB) and alert description text (BNJ92UTB).

3 A semicolon, indicating end of text, is required.

You can insert comment lines anywhere in the table by placing an asterisk in column 1. The remainder of that line serves as the comment.

4 Figure 26 on page 69 shows code point 0320 which is outside the reserved user code point range. You may want to use this 0320 in case it becomes an IBM-supplied code point after this NetView release. If the NetView program adds support, the code point text in the user table would be overridden by the code point text in the NetView table.

5 The imbed flag, shown as a y in column 6, indicates that a qualifier flag is turned on when the data associated with X'82' or X'83' subfield is to be placed before the code point's text, imbedded within the code point's text, or is following on the same line immediately after the code point's text. One blank must follow the imbed flag. Any character other than y indicates the imbed flag is off. If the imbed flag is turned on, the information is imbedded at the point marked by a dollar sign ($). Since no variable substitution can be done for probable cause and alert description, an imbed flag is ignored in BNJ93UTB and BNJ92UTB.

Link-Editing the User Tables

For MVS:

- CNMSJM07 parses the user tables for validity and link-edits them into a user-defined load data set. This data set should be concatenated in the STEPLIB of the NetView start procedure before SYS1.NPDALIB.

  Refer to *Network Program Products Samples* for an explanation of the parameters and execution of CNMSJM07.

- If no errors are found, the user table is link-edited into the data set defined to NETLNK in CNMSJM07. If errors are found, the job ends with a nonzero return code. A list of the errors is written to SYSPRINT.

For VM:

- CNMSVM07 parses the user tables for validity and link-edits them into a user-defined LOADLIB. This LOADLIB should be ahead of the NPDA LOADLIB in the GLOBAL LOADLIB statement in the NETSTRT GCS file.

  Refer to *Network Program Products Samples* for an explanation of the parameters and execution of CNMSVM07.

- If no errors are found, the user table is link-edited into the load library defined in CNMSVM07. If errors are found, the exec ends with a nonzero return code. A list of the errors is written to SYSPRINT.

Note: The NetView program recognizes only user-defined code points available at initialization. If additional code points are added while the NetView program is running, stop and restart the NetView program so the new code points will be recognized.

Adding or Modifying Resource Types

You can add new resource types for hierarchical displays in the hardware monitor by modifying the member BNJRESTY.

- **For MVS**, BNJRESTY is a member of the data set SYS1.BNJPNL2, defined by the definition statement BNJPNL2 in the NetView start procedure.
- **For VM**, BNJRESTY is a CMS file with a filetype of NCCFLST.

Figure 27 shows the format for BNJRESTY. The numerical references are explained after the figure.

```
1  2 3
10 DISK your comments
```

Figure 27. Sample Contents of BNJRESTY

1 A 2-character hexadecimal number, starting in column 1, flows to the NetView program in the X'05' subvector. Valid characters are 0 through 9 and A through F. If you include duplicate hexadecimal codes, the system uses the first entry for that code in BNJRESTY.

2 The four characters in columns 4 through 7 are taken as the resource type. Valid characters are 0 through 9, A through Z, and any printable special characters. A resource type of fewer than four characters must be padded on the right with blanks. Delimiters such as a comma (,), period (.), or equal sign (=) should not be used as characters in the resource type.

3 An optional comment can begin anywhere after the resource type.

If you modify BNJRESTY while the hardware monitor task BNJDSERV is active, the new resource types will not be recognized. You must use STOP TASK followed by STARTCNM NPDA to cause the NetView program to recognize any new resource types.

If the NetView program finds an invalid entry in BNJRESTY during activation of BNJDSERV, an error message appears on the command facility console and IBM-supplied resource types are used instead.

Chapter 5. Modifying SPCS and NAM Command Lists

This chapter describes service point command service (SPCS) commands and how to modify SPCS command lists. It also describes network asset management (NAM) command lists.

Service Point Command Service

The SPCS is a set of commands that support and enhance the NetView program's control of service points such as the NetView/PC interface program. A service point application manages non-SNA devices, such as front-end line switches and multiplexers. You can send commands to the service point application to do problem determination for these devices.

Four NetView SPCS commands can be used with service points for problem determination (see *NetView Operation* for the command format):

- LINKTEST—requests that the service point test a given link or link segment.
- LINKDATA—requests that the service point return device data for a given link or link segment.
- LINKPD—requests problem determination analysis from the service point on a given link or link segment.
- RUNCMD—sends service point application commands to the service point applications from the NetView program.

The SPCS commands are long-running commands that suspend the command list when they are executed. The command list resumes when the SPCS command is completed. After the command is completed, a return code is set. See *NetView Customization: Writing Command Lists* for more information.

Using SPCS Commands

The NetView program provides six command lists which issue SPCS commands; they may give you ideas on how to use SPCS commands from command lists for automation. You can modify these command lists for your particular application or use them as ideas for your own command lists. Following are the names and descriptions of each of the six command lists.

INITCNFG Contains the service point resource information. This command list will not work if any lowercase values are used. The configuration defined in INITCNFG should match the configuration of the lines controlled by your service points. If you invoke the INITCNFG command list from your NCCFIC command list, these global variables will be set up every time the NetView program is started. You can modify the following fields to contain the configuration of all the lines that you will be controlling with the SPCS commands.

LINE	line name
SP	service point name
APPL	application name
UN	using node name
RD	remote device name

Enter BROWSE INITCNFG to see this command list online.

ADDLINE Allows you to define common global variables for a new line to the SPLOOKUP command list. The syntax is:

```
ADDLINE LINE, SP, APPL, UN, RD
```

(UN and RD are not required. LINE, SP, and APPL are required.) Enter BROWSE ADDLINE to see this command list online.

SPLOOKUP Allows you to issue SPCS commands for a given line. SPLOOKUP determines the parameters needed to issue an SPCS command for the line from the global variables set up by INITCNFG. If SPLOOKUP is invoked with the LINE option, it returns the SP name, the APPL name, using node, and remote device for the specified line in task global variables SP, APPL, UN, and RD. If invoked with the SP option, SPLOOKUP returns a list of lines defined to the specified SP name in the task global variables LINECNT, LINE1, LINE2, and so forth. The syntax is:

```
SPLOOKUP LINE linename
```

```
SPLOOKUP SP spname
```

Enter BROWSE SPLOOKUP to see this command list online.

FINDNCP Displays events received by the NetView program from the using node for a service point. This command list is useful if you are having problems getting data back from a service point. FINDNCP looks up the using nodes for a given SP. If multiple using nodes are defined, they are listed. If only one using node is defined, FINDNCP invokes the NPDA Most Recent Events panel to show any events for the using node. Normally, the using node is the NCP linked to the SP. The syntax is:

```
FINDNCP spname
```

Enter BROWSE FINDNCP to see this command list online.

TESTSP Is used to issue SPCS commands for a given line. TESTSP invokes SPLOOKUP and then issues the SPCS commands and displays the results. You can use TESTSP as a base for command lists that execute SPCS commands automatically. It also demonstrates how to display results of the LINKTEST and LINKDATA commands. The syntax is:

```
TESTSP linename
```

Enter BROWSE TESTSP to see this command list online.

TESTRCMD Demonstrates how to use RUNCMD with the CLISTVAR keyword. It issues RUNCMD to execute a command at the service point. Results are stored in command list variables, and displays the data when control is returned.

The syntax is:

```
TESTRCMD sp appl 'command to execute at the service point'
```

(The command must be in single quotes.) Enter BROWSE TESTRCMD to see this command list online.

NAM Command Lists

Network asset management (NAM) provides an automated way of collecting inventory data from a subset of hardware and software devices. You can use NAM to collect vital product data (VPD) such as serial numbers, machine types, and model numbers for hardware products and software information, such as version and release level. However, the NetView program does not verify the returned data from devices supporting NAM. It only provides a way to collect and log the data. See *NetView Administration Reference* for information on the record formats. See *NetView Operation* for information on the IBM-supplied command lists.

Any device that supports the REQUEST/REPLY PSID and LPDA-2 architecture can report vital product data to the NetView program. (See *Systems Network Architecture Product Formats.*) An attempt to solicit vital product data from a device that does not support the architecture may cause the keyboard to lock or extraneous data to appear on the screen. You may need to press the reset key or clear the screen. This does not affect vital product data collection in the NetView program.

Following are some examples of physical units (PU) which support the REQUEST/REPLY PSID architecture.

- 3720/NCP, 3725/NCP, 3745/NCP
- 3174 — Reports data for itself and many types of attached devices such as:
  - Various models of 3191, 3192, and 3194 displays
  - Various models of PS/2, PC/AT, and PC/XT (OS/2-EE required).

  (See the user's guides, including the OS/2 EE user's guide, for each of the previous devices for instructions on entering vital product data into the device.)

Following are some examples of data communications equipment (DCE) which support the LPDA-2 modem and line status architecture.

- 586X modems
- 5822 DSU/CSU
- 7865/7825
- 7861/7868

The following software is required to support vital product data collection.

- VTAM V3R1.1 (with PTF UT25170) and subsequent releases
- NCP V4R3 and subsequent releases (hardware and software data)—A communications controller that is running NCP V4R3 reports both hardware and software information. (NCP V4R2 (software data only) — A communications controller that is running NCP V4R2 only reports data about the software it is running.)
- 3174 R4 microcode.

NAM provides the VPDCMD command to solicit vital product data from a given device, and the VPDLOG command to build and log a record to an external logging facility (SMF for MVS or the external log for VM). You may want to use Service Level Reporter (SLR) to view the data interactively or generate reports. You may use the

VPDALL command to generate VPDPU and VPDDCE command entries for all devices within a NetView domain. (See *NetView Operation* for information on the vital product data commands.) If you have any resources that require switched lines, be sure those lines are up before collecting vital product data.

NAM provides the following command lists:

VPDPU Collects and logs vital product data from a single PU and its attached devices. You can enter this command list from an operator's console or from another command list. See *NetView Administration Reference* for the record formats and *NetView Operation* for a description of VPDPU.

VPDDCE Solicits and logs vital product data from DCEs that are in a direct path between a specified NCP and a specified PU. You can issue this command list from an operator's console or from within another command list. See *NetView Administration Reference* for the record formats and *NetView Operation* for a description of VPDDCE.

VPDACT Is the default name of a command list that the VPDALL command generates when issued with the CREATE option. VPDALL reads a VTAM configuration member in VTAMLST as input and generates a command list called VPDACT (the default). VPDACT contains a list of VPDPU and VPDDCE entries for devices in your domain. VPDACT can be issued later by an operator to collect and to log vital product data from supported devices in the NetView domain.

VPDLOGC Is the command list that builds and logs START and END records. A START record is generated for a VPDACT command list at the beginning of a vital product data solicitation. An END record is generated for a VPDACT command list at the end of a vital product data solicitation. See *NetView Administration Reference* for the record formats. This command list should not be issued from an operator's console nor from within a user-written command list.

VPDXDOM Is a service command list used for vital product data solicitation from cross-domain resources. This command list is driven through a message automation table and should not be issued from an operator's console nor from within a user-written command list.

Vital Product Data Collection from a Single Physical Unit

The following example describes collecting vital product data from a single PU and its attached devices.

1. Issue VPDPU or VPDDCE command list with a resource name specified.
2. The command list issues a VPDCMD to solicit data from the specified resource and waits for response messages.
3. A PU responds with vital product data for itself or for itself and its attached devices.
4. The command list traps the response messages and saves the vital product data, such as machine type, model number, serial numbers, in command list variables.
5. When the completion message is received, the command list builds records and writes them to an external logging facility.
6. If any abnormal events (such as a SMF logging failure, VPDTASK is inactive, an ABEND) occur before completion, a command list error message is issued and the command list terminates.

Vital Product Data Collection from a Single NetView Domain

The following example describes collecting vital product data from a single NetView domain.

1. A NetView operator enters the following command:

   ```
   VPDALL CONFIG(ATCCON01),CREATE,CLIST(VPDACT),ADD
   ```

2. VPDALL reads the specified nodes from the configuration member (ATCCON01 in this example) in VTAMLST. VPDALL extracts from the VTAMLST nodes all the resource names for which vital product data is to be collected. It then builds VPDPU and VPDDCE entries in a command list called VPDACT. VPDALL does not support DCEs on switched lines nor dynamic reconfiguration decks (DRDS).

   Note: In order to collect data from your entire domain, the configuration member must contain the definitions for all the resources in your domain.
3. You can modify VPDACT by adding or deleting resource names.
4. When the VPDACT command list is executed, it calls VPDLOGC to generate a START record. It then calls the VPDPU and VPDDCE command lists. When the VPDPU and VPDDCE command lists have completed, VPDACT again calls VPDLOGC to generate an END record.

Focal Point Vital Product Data Collection

Figure 28 illustrates a focal point NetView for vital product data collection and is described in the example following the figure.

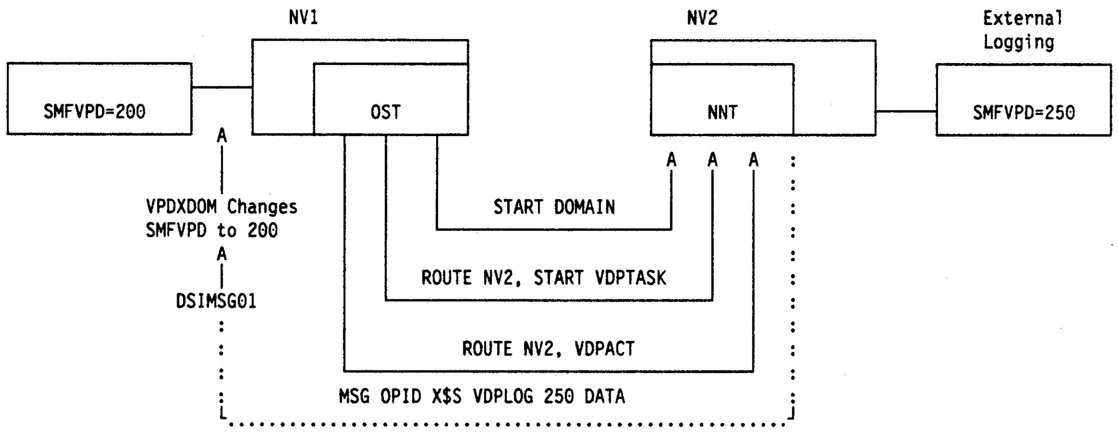


Figure 28. Vital Product Data Focal Point NetView

The following example describes collecting vital product data at a focal point NetView.

1. During installation, **NV1** sets the common global variable SMFVPD to 200. **NV2** sets it to 250.

   Note: The NetView initial command list (CNME1034) sets the common global variable SMFVPD to 37.
2. Also during installation, **NV1** is designated as a focal point NetView for vital product data collection. In the message automation table (DSIMSG01), for **NV1** only, uncomment the statement to drive the VPDXDOM command list. See *NetView Installation and Administration Guide* for information on DSIMSG01.
3. Start DSIELTSK from the focal point NetView **NV1**.
4. **NV1** (the focal point) establishes a direct OST to NNT session with **NV2** via the START DOMAIN command.

5. **NV1** (the focal point) issues START VPDTASK.
6. **NV1** (the focal point) issues ROUTE **NV2**, START VPDTASK.
7. **NV1** (the focal point) issues ROUTE **NV2**, VPDACT (for example). This causes the VPDACT command list in **NV2** to run under an NNT.
8. In **NV2**, VPDACT verifies that it is running under the NNT and then generates the following message:

   ```
   MSG OPID X$S VPDLOG 250 '1 STRING1 10 STRING2...'
   ```

   (X$S is a special string that is recognized by the message automation table.)
9. When the VPDACT command list in **NV2** writes the generated message to the operator in **NV1**, it triggers the message automation table. The message automation table executes the VPDXDOM command list in **NV1**.
10. When VPDXDOM is entered, the message string looks like this

    ```
    DSI039I MSG FROM OPID : X$S VPDLOG 250 1 STRING1...
    ```

11. VPDXDOM verifies that **NV1** has set SMFVPD as a common global variable. Since it has, the command list changes SMFVPD from 250 (**NV2**) to 200 (**NV1**).
12. VPDLOG executes and the data is logged under **NV1**'s SMF record number 200.
13. Be sure the cross-domain session remains established until the vital product data solicitation is completed for that session.

Customization Considerations

You may want to customize the IBM-supplied vital product data command lists to suit the needs of your installation.

When modifying NAM command lists to build different record formats, do not exceed 256 bytes per record. The NetView program has a command string limitation of 240 characters. Therefore, you may need to write a command processor to make full use of the VPD command. See *NetView Customization: Using Assembler* for information on command processors. If you are changing the SMF record format, you can no longer use the record number 37. You must globally define the SMF record number within the user-defined range of 128 to 255. If you are using SLR, the SLR table must be written to match your modified SMF record format.

See *NetView Operation* and *NetView Customization: Writing Command Lists* for limitations concerning the use of &WAIT and RESET, and the issuance of a second NAM command list and NAM command while a NAM command list is running.

To try increasing performance,

- You can write a command list that reads in VPDACT to distribute the workload among several autotasks. Dividing the workload among several OSTs or autotasks allows multiple VPDPU or VPDDCE entries to execute simultaneously. Otherwise, the VPDPU and VPDDCE entries are executed serially.
- Create several configuration members.
  - For example, one member per major node.
  - Using VPDALL, create several command lists.
  - Each command list can be run under several tasks, such as an OST and an autotask.

Appendix A. Color Maps for Hardware Monitor Panels

The following table lists the display name, display number, and color map for hardware monitor panels. See Chapter 4, "Customizing Hardware Monitor Displayed Data" on page 45 for more information on color maps.

Table 9 (Page 1 of 4). Color Maps for Hardware Monitor Displays

Display Name	Display Number	Color Map
Alerts-Dynamic	NPDA-30A	BNJMP30A
Alerts-Dynamic, rolling	NPDA-30A	BNJMP30R
Alerts-History	NPDA-31A	BNJMP31A
Alerts-Static	NPDA-30B	BNJMP30B
Command Description	NPDA-31N2	BNJMP11NB
Command Description Alerts-Dynamic	NPDA-11A	BNJMP11A
Command Description Controller	NPDA-11M	BNJMP11M
Command Description Copy	NPDA-11S	BNJMP11S
Command Description Display Domain	NPDA-11K	BNJMP11K
Command Description Display Ratio	NPDA-11Q	BNJMP11Q
Command Description Display Wrap Count	NPDA-11F	BNJMP11F
Command Description Filter Status	NPDA-11I	BNJMP11I
Command Description Most Recent	NPDA-11D	BNJMP11D
Command Description Most Recent, alternate	NPDA-11DB	BNJMP11DB
Command Description Purge Attached EV/ST Data	NPDA-11N3, page2	BNJMP1NC
Command Description Purge EV/ST Data	NPDA-11N1, page 1	BNJMP1NA
Command Description REPORTS	NPDA-11R	BNJMP11R
Command Description Set Domain	NPDA-11J	BNJMP11J
Command Description Set Ratio	NPDA-11P	BNJMP11P
Command Description Set Wrap Count	NPDA-11E	BNJMP11E
Command Description SRFILTER	NPDA-11GA	BNJMP1GA
Command Description SRFILTER	NPDA-11GB	BNJMP1GB
Command Description SRFILTER	NPDA-11GC	BNJMP1GC
Command Description SRFILTER	NPDA-11GD	BNJMP1GD
Command Description SRFILTER	NPDA-11GE	BNJMP1GE
Command Description SVFILTER	NPDA-11HA	BNJMP1HA
Command Description SVFILTER	NPDA-11HB	BNJMP1HB
Command Description SVFILTER	NPDA-11HC	BNJMP1HC
Command Description TEST	NPDA-11L	BNJMP11L
Command Description TOTAL	NPDA-11C	BNJMP11C
Command Description TOTAL, alternate	NPDA-11CB	BNJMP1CB
Command List	NPDA-10AA, page 1	BNJMP10A
Command List	NPDA-10AB, page 2	BNJMP10B
Common Format Glossary	NPDA-02C	BNJMP2C1
Controller Information Display	NPDA-02E	BNJMP02E
Controller (CTRL) Selection Menu	NPDA-CTRL	BNJMPCTL
Downstream Member of Token-Ring LAN Fault Domain	NPDA-44B	BNJMP4BH
DSU/CSU and Line Status DSU/CSU and Line Parameters Link Segment Level *n*	NPDA-22C, page 1	BNJMPDL1
DSU/CSU and Line Status Remote DSU/CSU Interface-Remote Device Status-Link Segment Level *n*	NPDA-22C, page 2	BNJMPDL2

Table 9 (Page 2 of 4). Color Maps for Hardware Monitor Displays

Display Name	Display Number	Color Map
DSU/CSU and Line Status Configuration Summary, Link Segment Level *n*	NPDA-22C, page 3	BNJMPDL3
Event Detail	NPDA-43B	BNJMP43B
Event Detail	NPDA-43M	BNJMP43M
Event Detail	NPDA-43N, 43Q	BNJMP43N
Event Detail	NPDA-43C	BNJMP43C
Event Detail	NPDA-43T	BNJMP43T
Event Detail	NPDA-43A	BNJMP43A
Event Detail	NPDA-43P	BNJMP43P
Event Detail	NPDA-43S	BNJMP43S
Event Detail, alternate	NPDA-43T	BNJMP434
Event Detail, alternate	NPDA-43S	BNJMP433
Event Detail for BSC Line	NPDA-43T	BNJMP43T
Event Detail for BSC Station	NPDA-43T	BNJMP43T
Event Detail for BSC/SS Line	NPDA-43B	BNJMP43B
Event Detail for BSC/SS Station	NPDA-43B	BNJMP43B
Event Detail for Channel-Attached Station	NPDA-43B	BNJMP43B
Event Detail for Channel Link	NPDA-43B	BNJMP43B
Event Detail for Instruction Exception	NPDA-43J	BNJMP43J
Event Detail for Miscellaneous Interrupts	NPDA-43K	BNJMP43D
Event Detail for Scanner-Type 1/4	NPDA-43G	BNJMP43D
Event Detail for Scanner-Type 2/3	NPDA-43H	BNJMP43D
Event Detail for Scanner-Type 1	NPDA-43D	BNJMP43D
Event Detail for Scanner-Type 2	NPDA-43E	BNJMP43D
Event Detail for Scanner-Type 3	NPDA-43F	BNJMP43D
Event Detail for Scanner-Type 4	NPDA-43I	BNJMP43D
Event Detail for SDLC Line	NPDA-43P	BNJMP43B
Event Detail for SDLC Line	NPDA-43T	BNJMP43T
Event Detail for SDLC Station	NPDA-43B	BNJMP43B
Event Detail for SDLC Station	NPDA-43T	BNJMP43T
Event Detail for 3270 Non-SNA Controller	NPDA-43L	BNJMP43L
Event Detail Menu	NPDA-43R	BNJMP43R
Event Detail Menu	NPDA-43R	BNJMP43R
Event Detail Menu, alternate	NPDA-43R	BNJMP432
Event Detail Menu for BSC Line	NPDA-43R	BNJMP43R
Event Detail Menu for BSC Line, alternate	NPDA-43T	BNJMP434
Event Detail Menu for BSC Station	NPDA-43R	BNJMP43R
Event Detail Menu for BSC Station, alternate	NPDA-43T	BNJMP434
Event Detail Menu for SDLC Line	NPDA-43R	BNJMP43R
Event Detail Menu for SDLC Line, alternate	NPDA-43T	BNJMP434
Event Detail Menu for SDLC Station	NPDA-43R	BNJMP43R
Event Detail Menu for SDLC Station, alternate	NPDA-43T	BNJMP434
Event Summary	NPDA-42A	BNJMP42A
Event Summary	NPDA-42B	BNJMP42B
Event Summary	NPDA-42C	BNJMP42C
Glossary displays	(many displays)	BNJMPGLO
Hexadecimal Display of Error Record	NPDA-44C	BNJMP44C
HELP Menu	NPDA-02B	BNJMP02B
Line Analysis-Link Segment Level *n*	NPDA-24B	BNJMPLNA
Link Configuration	NPDA-44A1	BNJMP441
Link Configuration	NPDA-44A2	BNJMP442
Link Configuration, alternate	NPDA-44A1	BNJMP443

Table 9 (Page 3 of 4). Color Maps for Hardware Monitor Displays

Display Name	Display Number	Color Map
Link Configuration Summary-Level Selection	NPDA-LSLS	BNJMPLSL
Link Data for SNA Controller	NPDA-23A	BNJMP23A
Link Problem Determination Aid (LPDA-1)Data	NPDA-52A	BNJMP52A
Link Problem Determination Aid (LPDA-1) LDM Data	NPDA-52AL	BNJMP52L
Link Problem Determination Aid (LPDA-2) Data Link Segment Level 1	NPDA-52B	BNJMP52B
Link Problem Determination Aid (LPDA-2) Data Link Segment Level 1, alternate	NPDA-52B	BNJMP522
Link Problem Determination Aid (LPDA) Data Link Segment Level 2	NPDA-52C	BNJMP52B
Link Status and Test Results	NPDA-24A	BNJMP24A
Link Status and Test Results for LDM	NPDA-24AL	BNJMP24L
LPDA-1 Command Menu	NPDA-LPDA1	BNJMPLP1
LPDA-2 Command Menu	NPDA-LPDA2	BNJMPLP2
Menu	NPDA-01A	BNJMP01A
Modem and Line Status Modem and Line Parameters Link Segment Level *n*	NPDA-22B, page 1	BNJMPML1
Modem and Line Status Remote Modem Interface-Remote Device Status-Link Segment Level *n*	NPDA-22B, page 2	BNJMPML2
Modem and Line Status Configuration Summary, Link Segment Level *n*	NPDA-22B, page 3	BNJMPML3
Most Recent Events	NPDA-41A	BNJMP41A
Most Recent Statistical Data	NPDA-51E	BNJMP51E
Most Recent Statistical Data	NPDA-51F	BNJMP51F
Most Recent Statistical Data	NPDA-51G	BNJMP51G
Most Recent Statistical Data	NPDA-51H	BNJMP51H
Most Recent Statistical Data	NPDA-51B	BNJMP51B
Most Recent Statistical Data for Printer	NPDA-51D	BNJMP51B
Most Recent Statistical Data for Tape	NPDA-51C	BNJMP51B
Most Recent Traffic Statistics	NPDA-51A	BNJMP51A
Most Recent Traffic Stats for BSC/SS Station	NPDA-51A	BNJMP51A
Most Recent Traffic Stats for BSC STA. w/LPDA	NPDA-51A	BNJMP51A
Most Recent Traffic Stats for Channel Attached STA.	NPDA-51A	BNJMP51A
Most Recent Traffic Stats for Local CTRL	NPDA-51A	BNJMP51A
Most Recent Traffic Stats for SDLC Station	NPDA-51A	BNJMP51A
Most Recent Traffic Stats for SDLC STA. w/LPDA	NPDA-51A	BNJMP51A
Multiple Entries for Selected Resource	NPDA-70A	BNJMP70A
Overwrite Map	(all displays)	BNJOVERW
Recommended Action for Selected Event	NPDA-BNIxxxyyy	BNJMP45A
Recording and Viewing Filter Status	NPDA-20A,20B	BNJMP20A
Release Level for SNA Controller	NPDA-21A	BNJMP21A
Remote DTE Interface Status	NPDA-25A	BNJMP25A
Remote DTE Interface Status for LDM	NPDA-25AL	BNJMP25A

Table 9 (Page 4 of 4). Color Maps for Hardware Monitor Displays

Display Name	**Display Number**	**Color Map**
Remote Self-Test Results	NPDA-22A	BNJMP22A
Remote Self-Test Results for LDM	NPDA-22AL	BNJMP22L
Reported Resource Hardware	NPDA-44B	BNJMP44B
Reported Resource Software Product	NPDA-44B	BNJMP4BS
Screen Control/Help	NPDA-02A, page 1	BNJMP2A1
Screen Control/Help	NPDA-02A, page 2	BNJMP2A2
Sender Hardware Product ID	NPDA-44B	BNJMP4BH
Sender Software Product ID	NPDA-44B	BNJMP4BS
Statistical Detail	NPDA-53E	BNJMP53E
Statistical Detail	NPDA-53F	BNJMP53F
Statistical Detail Menu	NPDA-53R	BNJMP43R
Statistical Detail Menu for BSC	NPDA-53R	BNJMP43R
Statistical Detail Menu for SDLC	NPDA-53R	BNJMP43R
TEST Information Display	NPDA-02D	BNJMP02D
Total Events	NPDA-40A	BNJMP40A
Total Statistical Data	NPDA-50A	BNJMP50A
Transmit Receive Test-Link Segment Level *n*	NPDA-25B	BNJMPTRT
Upstream Member of Token-Ring Fault Domain	NPDA-44B	BNJMP4BH

Appendix B. Programming-Interface Macros

The macros identified in this appendix are provided to allow a customer installation to write programs that use the services of NetView. Only those macros identified in this appendix should be used to request or receive the services of NetView.

All of the following macros are provided as product-sensitive programming interfaces. Product-sensitive programming interfaces are defined in "Special Notices" on page iii.

Name	Use
DSICBS	Control Block Services
DSICES	Command Entry Services
DSIDATIM	Date and Time
DSIDEL	Delete User-Defined Module
DSIDKS	Disk Services
DSIFIND	Find Long Running Command Storage
DSIFRE	Free Storage
DSIGET	Get Storage
DSIKVS	Keyword/Value Services
DSILCS	Obtain/Release Control Blocks
DSILOD	Load User-Defined Module
DSIMBS	Message Buffer Services
DSIMDS	Message Definition Services
DSIMQS	Message Queuing Services
DSIOIS	Operator Identification Services
DSIPAS	Parameter/Alias Services
DSIPOP	Remove Long Running Command
DSIPOS	ECB Post Services
DSIPRS	Parsing Services
DSIPSS	Presentation Services
DSIPUSH	Establish Long Running Command
DSIRDS	Resource Definition Services
DSIRXCOM	Access REXX Variables
DSIRXEBS	Get an EVALBLOK
DSISSS	Search Span Name Table Services
DSISYS	Operating System Indicator
DSIWAT	ECB Wait Services
DSIWCS	Write Console Services

DSIWLS	Write Log Services
DSIZCSMS	CNM Data Services
DSIZVSMS	VSAM Data Services

Appendix C. Programming-Interface Control Blocks and Include Files

Some control blocks and include files are provided as general-use programming interfaces, and some control blocks are provided as product-sensitive programming interfaces. General-use programming interfaces and product-sensitive programming interfaces are defined in "Special Notices" on page iii.

Control Blocks and Include Files That Are General-Use Programming Interfaces

The following control blocks and include files are provided as general-use programming interfaces:

Name	**Use**
DSIC	Main HLL C Include File
DSICCALL	HLL C Service Routine Definitions
DSICCNM	HLL C Return Codes
DSICCONS	HLL C Constants
DSICHLB	HLL C Mapping of DSIHLB
DSICORIG	HLL C Origin Block Mapping
DSICVARC	HLL C Varying Length Character Strings
DSIPCNM	HLL PL/I Return Codes
DSIPCONS	HLL PL/I Constants
DSIPHLB	HLL PL/I Mapping of DSIHLB
DSIPHLLS	PL/I Definitions for HLL Service Routines
DSIPLI	Main HLL PL/I Include File
DSIPORIG	HLL PL/I Origin Block Mapping

Control Blocks That Are Product-Sensitive Programming Interfaces

The following control blocks are provided as product-sensitive programming interfaces:

Name	**Use**
DSIART	Authorization and Routing Table
DSICBH	Control Block Header
DSICWB	Command Work Block
DSIDSB	Data Service Block
DSIDSRB	Data Services Request Block
DSIELB	External Logging Block
DSIIFR	Internal Function Request

DSILOGDS	NetView Log DSECT
DSIMVT	Main Vector Table
DSIOIT	Operator ID Table
DSIPDB	Parse Descriptor Block
DSISCE	System Command Entry
DSISCT	System Command Table (include only)
DSISNT	Span Name Table
DSISVL	Service Routine Vector List (include only)
DSISWB	Service Work Block
DSITIB	Task Information Block
DSITVB	Task Vector Block
DSIUSE	User Exit Parameter List

Glossary

A

abend. Abnormal end of task.

abnormal end of task (abend). Termination of a task before its completion because of an error condition that cannot be resolved by recovery facilities while the task is executing.

ACB. (1) In VTAM, access method control block. (2) In NCP, adapter control block.

ACB name. (1) The name of an ACB macroinstruction. (2) A name specified in the ACBNAME parameter of a VTAM APPL statement. Contrast with *network name*.

accept. For a VTAM application program, to establish a session with a logical unit (LU) in response to a CINIT request from a system services control point (SSCP). The session-initiation request may begin when a terminal user logs on, a VTAM application program issues a macroinstruction, or a VTAM operator issues a command. See also *acquire (1)*.

access method. A technique for moving data between main storage and input/output devices.

access method control block (ACB). A control block that links an application program to VSAM or VTAM.

accounting exit routine. In VTAM, an optional installation exit routine that collects statistics about session initiation and termination.

ACF. Advanced Communications Function.

ACF/NCP. Advanced Communications Function for the Network Control Program. Synonym for *NCP*.

ACF/SSP. Advanced Communications Function for the System Support Programs. Synonym for *SSP*.

ACF/VTAM. Advanced Communications Function for the Virtual Telecommunications Access Method. Synonym for *VTAM*.

acquire. (1) For a VTAM application program, to initiate and establish a session with another logical unit (LU). The acquire process begins when the application program issues a macroinstruction. See also *accept*. (2) To take over resources that were formerly controlled by an access method in another domain, or to resume control of resources that were controlled by this domain but released. Contrast with *release*. See also *resource takeover*.

active. (1) The state a resource is in when it has been activated and is operational. Contrast with *inactive, pending,* and *inoperative*. (2) Pertaining to a major or minor node that has been activated by VTAM. Most resources are activated as part of VTAM start processing or as the result of a VARY ACT command.

adapter. Hardware card that allows a device, such as a PC, to communicate with another device, such as a monitor, a printer, or other I/O device.

adapter control block (ACB). In NCP, a control block that contains line control information and the states of I/O operations for BSC lines, SS lines, or SDLC links.

adaptive session pacing. Synonym for *adaptive session-level pacing*.

adaptive session-level pacing. A form of session-level pacing in which session components exchange pacing windows that may vary in size during the course of a session. This allows transmission to adapt dynamically to variations in availability and demand of buffers on a session by session basis. Session pacing occurs within independent stages along the session path according to local congestion at the intermediate nodes. Synonymous with *adaptive session pacing*. See *pacing*, *session-level pacing*, and *virtual route pacing*.

Advanced Communications Function (ACF). A group of IBM licensed programs (principally VTAM, TCAM, NCP, and SSP) that use the concepts of Systems Network Architecture (SNA), including distribution of function and resource sharing.

alert. (1) In SNA, a record sent to a system problem management focal point to communicate the existence of an alert condition. (2) In the NetView program, a high priority event that warrants immediate attention. This data base record is generated for certain event types that are defined by user-constructed filters.

alert sender. The SNA entity, either a physical unit (PU) or a control point, that sends or causes a formatted alert to be sent to a focal point.

alias name. A name defined in a host used to represent a logical unit name, logon mode table name, or class-of-service name in another network. This name is defined to a name translation program when the alias name does not match the real name. The alias name translation program is used to associate the real and alias names.

API. Application program interface.

application program. (1) A program written for or by a user that applies to the user's work. (2) A program used to connect and communicate with stations in a network, enabling users to perform application-oriented activities.

application program interface (API). (1) The formally defined programming language interface between an IBM system control program or licensed program and its user. (2) The interface through which an application program interacts with an access method. In VTAM, it is the language structure used in control blocks so that application programs can reference them and be identified to VTAM.

attaching device. Any device that is physically connected to a network and can communicate over the network.

authorization exit routine. In VTAM, an optional installation exit routine that approves or disapproves requests for session initiation.

automatic logon. (1) A process by which VTAM automatically creates a session-initiation request to establish a session between two logical units (LUs). The session will be between a designated primary logical unit (PLU) and a secondary logical unit (SLU) that is neither queued for nor in session with another PLU. See also *controlling application program* and *controlling logical unit.* (2) In VM, a process by which a virtual machine is initiated by other than the user of that virtual machine. For example, the primary VM operator's virtual machine is activated automatically during VM initialization.

available. In VTAM, pertaining to a logical unit that is active, connected, enabled, and not at its session limit.

B

basic conversation. A conversation that supports the functions of the basic conversation protocol boundary defined by LU 6.2. That format requires data to be sent as logical records consisting of a 2-byte length prefix followed by the data. See also *mapped conversation.*

binary synchronous communication (BSC). (1) Communication using binary synchronous line discipline. (2) A uniform procedure, using a standardized set of control characters and control character sequences, for synchronous transmission of binary-coded data between stations.

BIU segment. In SNA, the portion of a basic information unit (BIU) that is contained within a path information unit (PIU). It consists of either a request/response header (RH) followed by all or a portion of a request/response unit (RU), or only a portion of an RU.

blocking of PIUs. In SNA, an optional function of path control that combines multiple path information units (PIUs) into a single basic transmission unit (BTU).

boundary function. (1) A capability of a subarea node to provide protocol support for attached peripheral nodes, such as: (a) interconnecting subarea path control and peripheral path control elements, (b) performing session sequence numbering for low-function peripheral nodes, and (c) providing session-level pacing support. (2) The component that provides these capabilities. See also *boundary node*, *network addressable unit (NAU)*, *peripheral path control*, *subarea node*, and *subarea path control.*

boundary node. (1) A subarea node with boundary function. See *subarea node.* See also *boundary function.* (2) The programming component that performs FID2 (format identification type 2) conversion, channel data link control, pacing, and channel or device error recovery procedures for a locally attached station. These functions are similar to those performed by a network control program for an NCP-attached station.

browse. A way of looking at a file that does not allow you to change it.

BSC. Binary synchronous communication.

buffer. A portion of storage for temporarily holding input or output data.

C

call. (1) * (ISO) The action of bringing a computer program, a routine, or a subroutine into effect, usually by specifying the entry conditions and jumping to an entry point. (2) To transfer control to a procedure, program, routine, or subroutine. (3) The actions necessary to make a connection between two stations. (4) To attempt to contact a user, regardless of whether the attempt is successful.

call establishment. The complete sequence of events necessary to establish a data connection.

calling. * (ISO) The process of transmitting selection signals in order to establish a connection between data stations.

CALLOUT. The logical channel type on which the data terminal equipment (DTE) can send a call, but cannot receive one.

CDRM. Cross-domain resource manager.

channel. * A path along which signals can be sent, for example, data channel, output channel. See *data channel* and *input/output channel.* See also *link.*

channel adapter. A communication controller hardware unit used to attach the controller to a System/360 or a System/370 channel.

channel link. A System/370 I/O channel to control unit interface that has an SNA network address. A channel link can be either a subarea link or a peripheral link and is defined in an NCP generation definition using the GROUP, LINE, and PU definition statements. See also *link* and *subarea link*.·

channel-attached. (1) Pertaining to the attachment of devices directly by input/output channels to a host processor. (2)

Pertaining to devices attached to a controlling unit by cables, rather than by telecommunication lines. Contrast with *link-attached*. Synonymous with *local*.

circuit switching. (1) * (ISO) A process that, on demand, connects two or more data terminal equipments (DTEs) and permits the exclusive use of a data circuit between them until the connection is released. (2) Synonymous with *line switching*. (3) See also *message switching* and *packet switching*.

class of service (COS). In SNA, a designation of the path control network characteristics, such as path security, transmission priority, and bandwidth, that apply to a particular session. The end user designates class of service at session initiation by using a symbolic name that is mapped into a list of virtual routes, any one of which can be selected for the session to provide the requested level of service.

CMS. Conversational Monitor System.

CNM. Communication network management.

code point. In the NetView/PC program and in the NetView program, a 1- or 2-byte hexadecimal value that indexes a text string stored at an alert receiver and is used by the alert receiver to create displays of alert information.

command. (1) A request from a terminal for the performance of an operation or the execution of a particular program. (2) In SNA, any field set in the transmission header (TH), request header (RH), and sometimes portions of a request unit (RU), that initiates an action or that begins a protocol; for example: (a) Bind Session (session-control request unit), a command that activates an LU-LU session, (b) the change-direction indicator in the RH of the last RU of a chain, (c) the virtual route reset window indicator in a FID4 transmission header. See also *VTAM operator command*.

command facility. The component of the NetView program that is a base for command processors that can monitor, control, automate, and improve the operation of a network.

command list. A list of commands and statements designed to perform a specific function for the user. Command lists can be written in REXX or in NetView Command List Language.

command procedure. Either a command list or a command processor.

command processor. A user-written module designed to perform a specific function. Command processors, which can be written in assembler or a high-level language (HLL), are invoked as commands.

communication control unit. A communication device that controls the transmission of data over lines in a network. Communication control units include transmission control units (such as the 2702 Transmission Control Unit) and communication controllers (such as the 3720 or 3725).

communication line. Deprecated term for *telecommunication line* and *transmission line*.

communication management configuration host node. The type 5 host processor in a communication management configuration that does all network-control functions in the network except for the control of devices channel-attached to data hosts. Synonymous with *communication management host*. Contrast with *data host node*.

communication management host. Synonym for *communication management configuration host node*. Contrast with *data host*.

communication network management (CNM). The process of designing, installing, operating, and managing the distribution of information and controls among end users of communication systems.

communication network management (CNM) application program. A VTAM application program that issues and receives formatted management services request units for physical units. For example, the NetView program.

communication network management (CNM) interface. The interface that the access method provides to an application program for handling data and commands associated with communication system management. CNM data and commands are handled across this interface.

communication network management (CNM) processor. A program that manages one of the functions of a communications system. A CNM processor is executed under control of the NetView program.

component. Any part of a network other than an attaching device, such as an access unit.

composite end node (CEN). A group of nodes made up of a single type 5 node and its subordinate type 4 nodes that together support type 2.1 protocols. To a type 2.1 node, a CEN appears as one end node. For example, NCP and VTAM act as a composite end node.

configuration. (1) (TC97) The arrangement of a computer system or network as defined by the nature, number, and the chief characteristics of its functional units. The term may refer to a hardware or a software configuration. (2) The devices and programs that make up a system, subsystem, or network. (3) In CCP, the arrangement of controllers, lines, and terminals attached to an IBM 3710 Network Controller. Also, the collective set of item definitions that describe such a configuration.

configuration services. In SNA, one of the types of network services in the control point (CP) and in the physical unit (PU); configuration services activate, deactivate, and maintain the status of physical units, links, and link stations. Configuration services also shut down and restart network elements and modify path control routing tables and address-translation tables. See also *maintenance services*, *management services*, *network services*, and *session services*.

connected. In VTAM, pertaining to a physical unit (PU) or logical unit (LU) that has an active physical path to the host processor containing the system services control point (SSCP) that controls the PU or LU.

connection. Synonym for *physical connection*.

control block. (1) (ISO) A storage area used by a computer program to hold control information. (2) In the IBM Token-Ring Network, a specifically formatted block of information provided from the application program to the Adapter Support Interface to request an operation.

control point (CP). (1) A system services control point (SSCP) that provides hierarchical control of a group of nodes in a network. (2) A control point (CP) local to a specific node that provides control of that node, either in the absence of SSCP control (for type 2.1 nodes engaged in peer to peer communication) or to supplement SSCP control.

control program (CP). The VM operating system that manages the real processor's resources and is responsible for simulating System/370s for individual users.

controller. A unit that controls input/output operations for one or more devices.

controlling application program. In VTAM, an application program with which a secondary logical unit (other than an application program) is automatically put in session whenever the secondary logical unit is available. See also *automatic logon* and *controlling logical unit*.

controlling logical unit. In VTAM, a logical unit with which a secondary logical unit (other than an application program) is automatically put in session whenever the secondary logical unit is available. A controlling logical unit can be either an application program or a device-type logical unit. See also *automatic logon* and *controlling application program*.

conversation. In SNA, a logical connection between two transaction programs using an LU 6.2 session. Conversations are delimited by brackets to gain exclusive use of a session.

Conversational Monitor System (CMS). A VM application program for general interactive time sharing, problem solving, and program development.

CP. (1) Control program. (2) Control point.

cross-domain. In SNA, pertaining to control of resources involving more than one domain.

cross-domain resource (CDRSC). A resource owned by a cross-domain resource manager (CDRM) in another domain but known by the CDRM in this domain by network name and associated CDRM.

cross-domain resource manager (CDRM). In VTAM, the function in the system services control point (SSCP) that controls initiation and termination of cross-domain sessions.

D

DASD. Direct access storage device.

data channel. Synonym for *input/output channel*. See *channel*.

data circuit-terminating equipment (DCE). (TC97) The equipment installed at the user's premises that provides all functions required to establish, maintain, and terminate a connection, and the signal conversion and coding between the data terminal equipment (DTE) and the line. The DCE may be separate equipment or an integral part of other equipment.

data host. Synonym for *data host node*. Contrast with *communication management configuration host*.

data host node. In a communication management configuration, a type 5 host node that is dedicated to processing applications and does not control network resources, except for its channel-attached or communication adapter-attached devices. Synonymous with *data host*. Contrast with *communication management configuration host node*.

data link. In SNA, synonym for *link*.

data link control (DLC) layer. In SNA, the layer that consists of the link stations that schedule data transfer over a transmission medium connecting two nodes and perform error control for the link connection. Examples of data link control are SDLC for serial-by-bit link connection and data link control for the System/370 channel.

data services command processor (DSCP). A component that structures a request for recording and retrieving data in the application program's data base and for soliciting data from a device in the network.

data services task (DST). The NetView subtask that gathers, records, and manages data in a VSAM file and/or a network device that contains network management information.

data set. The major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access.

data terminal equipment (DTE). (TC97) That part of a data station that serves as a data source, data link, or both, and provides for the data communication control function according to protocols.

data types. In the NetView program, a concept to describe the organization of panels. Data types are defined as alerts, events, and statistics. Data types are combined with resource types and display types to describe NetView's display organization. See also *display types* and *resource types*.

DCE. Data circuit-terminating equipment.

ddname. Data definition name.

definite response (DR). In SNA, a value in the form-of-response-requested field of the request header. The value directs the receiver of the request to return a response unconditionally, whether positive or negative, to that request. Contrast with *exception response* and *no response*.

definition statement. (1) In VTAM, the statement that describes an element of the network. (2) In NCP, a type of instruction that defines a resource to the NCP. See Figure 29, Figure 30, and Figure 31. See also *macroinstruction*.

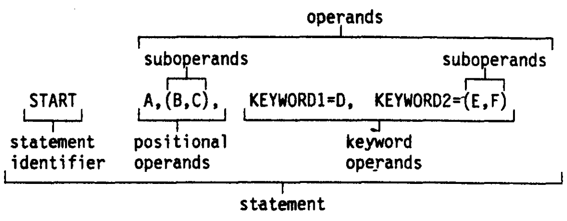


Figure 29. Example of a Language Statement

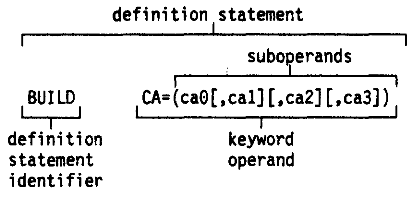
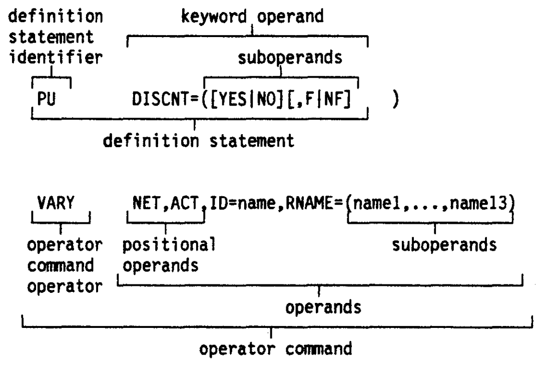


Figure 30. NCP Examples



definition statement identifier
keyword operand
suboperands
PU
DISCNT=([YES|NO][,F|NF])
definition statement
VARY
NET,ACT,ID=name,RNAME=(name1,...,name13)
operator command operator
positional operands
suboperands
operands
operator command

Figure 31. VTAM Examples

detailed data. Short strings of textual data transported in a network management vector transport (NMVT) and displayed, without any interpretation or translation, by a problem management focal-point program.

device. An input/output unit such as a terminal, display, or printer. See *attaching device*.

direct access storage device (DASD). A device in which the access time is effectively independent of the location of the data. For example, a disk.

directory. In VM, a control program (CP) disk that defines each virtual machine's normal configuration.

disabled. In VTAM, pertaining to a logical unit (LU) that has indicated to its system services control point (SSCP) that it is temporarily not ready to establish LU-LU sessions. An initiate request for a session with a disabled logical unit (LU) can specify that the session be queued by the SSCP until the LU becomes enabled. The LU can separately indicate whether this applies to its ability to act as a primary logical unit (PLU) or a secondary logical unit (SLU). See also *enabled* and *inhibited*.

display. (1) To present information for viewing, usually on a terminal screen or a hard-copy device. (2) A device or medium on which information is presented, such as a terminal screen. (3) Deprecated term for *panel*.

display levels. Synonym for *display types*.

display types. In the NetView program, a concept to describe the organization of panels. Display types are defined as total, most recent, user action, and detail. Display types are combined with resource types and data types to describe NetView's panel organization. See *data types* and *resource types*. Synonymous with *display levels*..

domain. (1) An access method, its application programs, communication controllers, connecting lines, modems, and attached terminals. (2) In SNA, a system services control point (SSCP) and the physical units (PUs), logical units (LUs), links, link stations, and all the associated resources that the SSCP has the ability to control by means of activation requests and deactivation requests. See *system services control point domain* and *type 2.1 node control point domain*. See also *single-domain network* and *multiple-domain network*.

domain operator. In a multiple-domain network, the person or program that controls the operation of the resources controlled by one system services control point. Contrast with *network operator* (2).

double-byte character set (DBCS). A character set, such as Kanji, in which each character is represented by a two-byte code.

downstream. In the direction of data flow from the host to the end user. Contrast with *upstream*.

DRDS. Dynamic reconfiguration data set.

DSCP. Data services command processor.

DST. Data services task.

DSU/CSU. Data service unit/channel service unit.

DTE. Data terminal equipment.

dump. (1) Computer printout of storage. (2) To write the contents of all or part of storage to an external medium as a safeguard against errors or in connection with debugging. (3) (ISO) Data that have been dumped.

dynamic reconfiguration (DR). The process of changing the network configuration (peripheral PUs and LUs) without regenerating complete configuration tables.

dynamic reconfiguration data set (DRDS). In VTAM, a data set used for storing definition data that can be applied to a generated communication controller configuration at the operator's request. A dynamic reconfiguration data set can be used to dynamically add PUs and LUs, delete PUs and LUs, and move PUs. It is activated with the VARY DRDS operator command. See also *dynamic reconfiguration*.

E

E/T. Error-to-traffic.

EBCDIC. * Extended binary-coded decimal interchange code. A coded character set consisting of 8-bit coded characters.

ECB. Event control block.

ED. Enciphered data.

element. (1) A field in the network address. (2) The particular resource within a subarea identified by the element address. See also *subarea*.

Emulation Program (EP). An IBM control program that allows a channel-attached 3705 or 3725 communication controller to emulate the functions of an IBM 2701 Data Adapter Unit, an IBM 2702 Transmission Control, or an IBM 2703 Transmission Control. See also *network control program*.

enabled. In VTAM, pertaining to a logical unit (LU) that has indicated to its system services control point (SSCP) that it is now ready to establish LU-LU sessions. The LU can separately indicate whether this prevents it from acting as a primary logical unit (PLU) or as a secondary logical unit (SLU). See also *disabled* and *inhibited*.

enciphered data (ED). Data whose meaning is concealed from unauthorized users.

end node. A type 2.1 node that does not provide any intermediate routing or session services to any other node. For example, APPC/PC is an end node. See *composite end node*, *node*, and *type 2.1 node*.

end user. In SNA, the ultimate source or destination of application data flowing through an SNA network. An end user may be an application program or a terminal operator.

entry point. An SNA node that provides distributed network management support. It may be a type 2, type 2.1, type 4, or type 5 node. It sends SNA-formatted network management data about itself and the resources it controls to a focal point for centralized processing, and it receives and executes focal point initiated commands to manage and control its resources.

EP. Emulation Program.

ER. (1) Explicit route. (2) Exception response.

error-to-traffic (E/T). The number of temporary errors compared to the traffic associated with a resource.

event. (1) In the NetView program, a record indicating irregularities of operation in physical elements of a network. (2) An occurrence of significance to a task; typically, the completion of an asynchronous operation, such as an input/output operation.

event control block (ECB). A control block used to represent the status of an event.

exception response (ER). In SNA, a value in the form-of-response-requested field of a request header (RH). An exception response is sent only if a request is unacceptable as received or cannot be processed. Contrast with *definite response* and *no response*. See also *negative response*.

EXEC. In a VM operating system, a user-written command file that contains CMS commands, other user-written commands, and execution control statements, such as branches.

exit routine. Any of several types of special-purpose user-written routines. See *accounting exit routine, authorization exit routine, logon-interpret routine, virtual route selection exit routine, EXLST exit routine*, and *RPL exit routine*.

EXLST exit routine. In VTAM, a routine whose address has been placed in an exit list (EXLST) control block. The addresses are placed there with the EXLST macroinstruction, and the routines are named according to their corresponding operand; hence DFASY exit routine, TPEND exit routine, RELREQ exit routine, and so forth. All exit list routines are coded by the VTAM application programmer. Contrast with *RPL exit routine*.

explicit route (ER). In SNA, the path control network elements, including a specific set of one or more transmission groups, that connect two subarea nodes. An explicit route is identified by an origin subarea address, a destination subarea address, an explicit route number, and a reverse explicit route number. Contrast with *virtual route (VR)*. See also *path* and *route extension*.

extended architecture (XA). An extension to System/370 architecture that takes advantage of continuing high performance enhancements to computer system hardware.

F

fault domain. In IBM Token-Ring Network problem determination, the portion of a ring that is involved with an indicated error.

feature. A particular part of an IBM product that a customer can order separately.

filter. In the NetView program, a function that limits the data that is to be recorded on the data base and displayed at the terminal. See *recording filter* and *viewing filter*.

flow control. In SNA, the process of managing the rate at which data traffic passes between components of the network. The purpose of flow control is to optimize the rate of flow of message units, with minimum congestion in the network; that is, to neither overflow the buffers at the receiver or at intermediate routing nodes, nor leave the receiver waiting for more message units. See also *adaptive session-level pacing, pacing, session-level pacing*, and *virtual route pacing*.

focal point. (1) An entry point that provides centralized management and control for other entry points for one or more network management categories. (2) In the NetView program, the focal point domain is the central host domain. It is the central control point for any management services element containing control of the network management data.

frame. (1) The unit of transmission in some local area networks, including the IBM Token-Ring Network. It includes delimiters, control characters, information, and checking characters. (2) In SDLC, the vehicle for every command, every response, and all information that is transmitted using SDLC procedures.

full-screen mode. A form of panel presentation in the NetView program where the contents of an entire terminal screen can be displayed at once. Full-screen mode can be used for fill-in-the-blanks prompting. Contrast with *line mode*.

G

GCS. Group control system.

generation. The process of assembling and link editing definition statements so that resources can be identified to all the necessary programs in a network.

generic alert. Encoded alert information that uses code points (defined by IBM and possibly customized by users or application programs) stored at an alert receiver, such as the NetView program.

group. In the NetView/PC program, to identify a set of application programs that are to run concurrently.

group control system (GCS). A component of VM that provides multiprogramming and shared memory support to virtual machines. It is a saved system intended for use with SNA products.

group control system group. A group of virtual machines that share common storage and load the same saved-VM system through a control program (CP) command or directory entry.

H

half-session. In SNA, a component that provides function management data (FMD) services, data flow control, and transmission control for one of the sessions of a network addressable unit (NAU). See also *primary half-session* and *secondary half-session*.

hard-copy task (HCT). The NetView subtask that controls the passage of data between the NetView program and the hard-copy device.

hardware monitor. The component of the NetView program that helps identify network problems, such as hardware, software, and microcode, from a central control point using interactive display techniques.

HCT. Hard-copy task.

help panel. An online display that tells you how to use a command or another aspect of a product. See *task panel*.

hierarchy. In the NetView program, the resource types, display types, and data types that make up the organization, or levels, in a network.

High Performance Option (HPO). A licensed program that is an extension of VM/SP. It provides performance and operation enhancements for large system environments. See *Virtual Machine/System Product High Performance Option*.

high-level language (HLL). A programming language that does not reflect the structure of any particular computer or operating system. For NetView Release 3, the high-level languages are PL/I and C.

HLL. High-level language.

host node. A node providing an application program interface (API) and a common application interface. See *boundary node*, *node*, *peripheral node*, *subarea host node*, and *subarea node*. See also *boundary function* and *node type*.

HPO. High Performance Option.

I

immediate command. In the NetView program, a command (such as GO, CANCEL, or RESET) that can be executed while a regular command is being processed.

inactive. Describes the state of a resource that has not been activated or for which the VARY INACT command has been issued. Contrast with *active*. See also *inoperative*.

information (I) format. A format used for information transfer.

inhibited. In VTAM, pertaining to a logical unit (LU) that has indicated to its system services control point (SSCP) that it is not ready to establish LU-LU sessions. An initiate request for a session with an inhibited LU will be rejected by the SSCP. The LU can separately indicate whether this applies to its ability to act as a primary logical unit (PLU) or as a secondary logical unit (SLU). See also *enabled* and *disabled*.

inoperative. The condition of a resource that has been active, but is not. The resource may have failed, received an INOP request, or is suspended while a reactivate command is being processed. See also *inactive*.

input/output channel. (1) (ISO) In a data processing system, a functional unit that handles the transfer of data between internal and peripheral equipment. (2) In a computing system, a functional unit, controlled by a processor, that handles the transfer of data between processor storage and local peripheral devices. Synonymous with *data channel*. See *channel*. See also *link*.

Interactive System Productivity Facility (ISPF). An IBM licensed program that serves as a full-screen editor and dialogue manager. Used for writing application programs, it provides a means of generating standard screen panels and interactive dialogues between the application programmer and terminal user.

interface. * A shared boundary. An interface might be a hardware component to link two devices or it might be a portion of storage or registers accessed by two or more computer programs.

ISPF. Interactive System Productivity Facility.

item. In CCP, any of the components, such as communication controllers, lines, cluster controllers, and terminals, that comprise an IBM 3710 Network Controller configuration.

J

JCL. Job control language.

job control language (JCL). * A problem-oriented language designed to express statements in a job that are used to identify the job or describe its requirements to an operating system.

K

Kanji. An ideographic character set used in Japanese. See also *double-byte character set*.

Katakana. A phonetic character set used in Japanese. It belongs to a single-byte character set.

keyword. (1) (TC97) A lexical unit that, in certain contexts, characterizes some language construction. (2) * One of the predefined words of an artificial language. (3) One of the significant and informative words in a title or document that describes the content of that document. (4) A name or symbol that identifies a parameter. (5) A part of a command operand that consists of a specific character string (such as DSNAME=). See also *definition statement* and *keyword operand*. Contrast with *positional operand*.

keyword operand. An operand that consists of a keyword followed by one or more values (such as DSNAME=HELLO). See also *definition statement*. Contrast with *positional operand*.

keyword parameter. A parameter that consists of a keyword followed by one or more values.

L

LAN. An industry-wide acronym for local area network.

line. See *communication line*.

line mode. A form of screen presentation in which the information is presented a line at a time in the message area of the terminal screen. Contrast with *full-screen mode*.

line switching. Synonym for *circuit switching*.

link. In SNA, the combination of the link connection and the link stations joining network nodes; for example: (1) a System/370 channel and its associated protocols, (2) a serial-by-bit connection under the control of Synchronous Data Link Control (SDLC). A link connection is the physical medium of transmission. A link, however, is both logical and physical. Synonymous with *data link*. See Figure 32 on page 96.

link connection segment. A portion of the configuration that is located between two resources listed consecutively in the service point command service (SPCS) query link configuration request list.

link problem determination aid (LPDA). A series of testing procedures initiated by the NetView program or NCP that provide modem status, attached device status, and the overall quality of a communications link.

link status (LS). Information maintained by local and remote modems.

link-attached. Pertaining to devices that are physically connected by a telecommunication line. Contrast with *channel-attached*. Synonymous with *remote*.

load module. (ISO) A program unit that is suitable for loading into main storage for execution; it is usually the output of a linkage editor.

local. Pertaining to a device that is attached to a controlling unit by cables, rather than by a telecommunication line. Synonymous with *channel-attached*.

local address. In SNA, an address used in a peripheral node in place of an SNA network address and transformed to or from an SNA network address by the boundary function in a subarea node.

local area network (LAN). (1) A network in which a set of devices are connected to one another for communication and that can be connected to a larger network. See also *token ring*. (2) A network in which communications are limited to a moderately sized geographic area such as a single office building, warehouse, or campus, and which do not generally extend across public rights-of-way. Contrast with *wide area network*.

logical record. (1) (TC97) A set of related data or words considered to be a record from a logical viewpoint. (2) A unit of information normally pertaining to a single subject; a logical record is that user record requested of or given to the data management function. See also *basic conversation*.

logical unit (LU). In SNA, a port through which an end user accesses the SNA network and the functions provided by system services control points (SSCPs). An LU can support at least two sessions—one with an SSCP and one with another LU—and may be capable of supporting many sessions with other LUs. See also *network addressable unit (NAU)*, *peripheral LU*, *physical unit (PU)*, *system services control point (SSCP)*, *primary logical unit (PLU)*, and *secondary logical unit (SLU)*.

logical unit (LU) services. In SNA, capabilities in a logical unit to: (1) receive requests from an end user and, in turn, issue requests to the system services control point (SSCP) in order to perform the requested functions, typically for session initiation; (2) receive

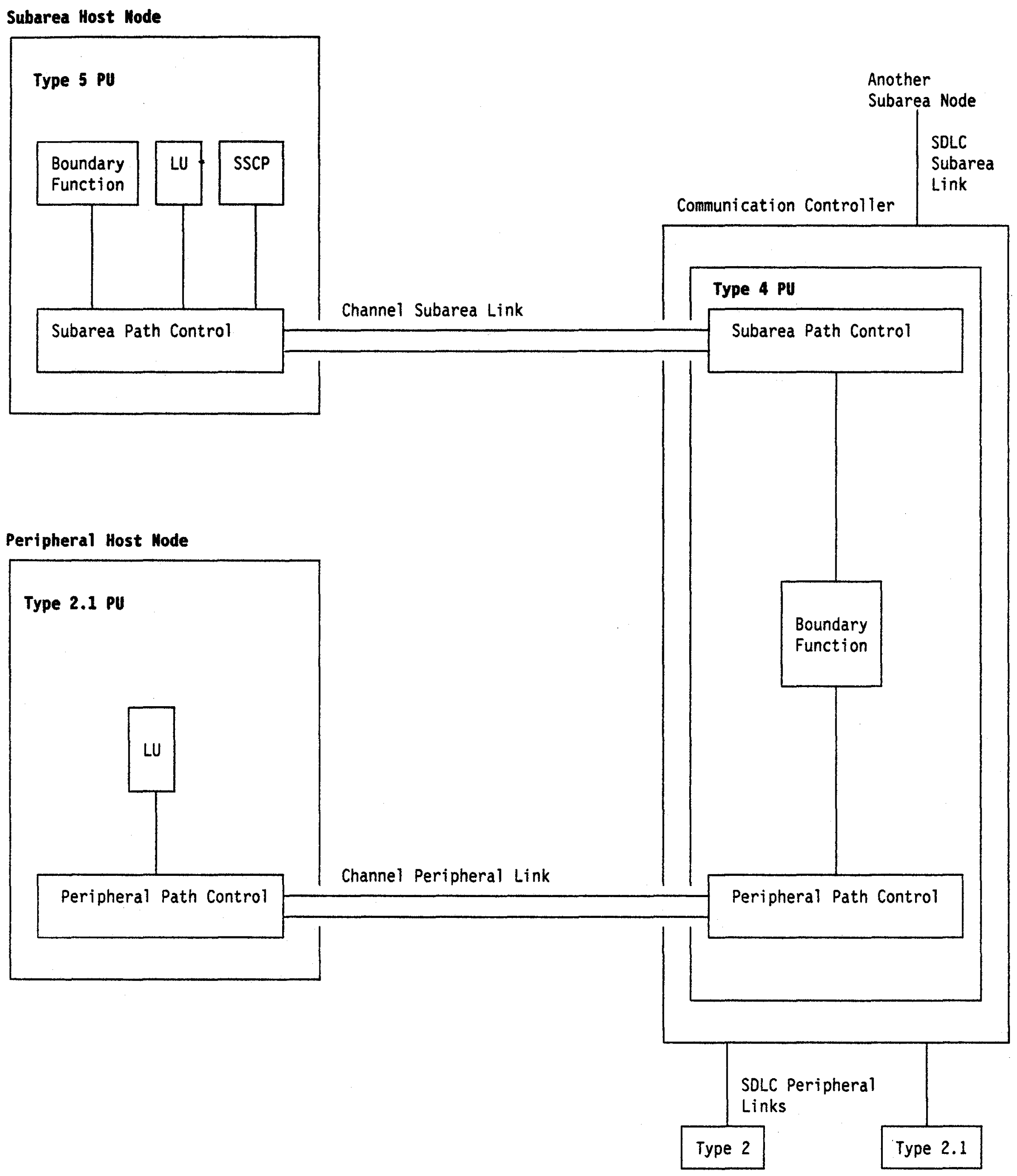


Figure 32. Links and Path Controls

requests from the SSCP, for example to activate LU-LU sessions via Bind Session requests; and (3) provide session presentation and other services for LU-LU sessions. See also *physical unit (PU) services.*

logical unit (LU) 6.2: A type of logical unit that supports general communication between programs in a distributed processing environment. LU 6.2 is characterized by (1) a peer relationship between session partners, (2) efficient utilization of a session for multiple transactions, (3) comprehensive end-to-end error processing, and (4) a generic application program interface (API) consisting of structured verbs that are mapped into a product implementation.

logmode table. Synonym for *logon mode table.*

logoff. In VTAM, an unformatted session termination request.

logon. In VTAM, an unformatted session initiation request for a session between two logical units. See *automatic logon* and *simulated logon*. See also *session-initiation request*.

logon mode. In VTAM, a subset of session parameters specified in a logon mode table for communication with a logical unit. See also *session parameters*.

logon mode table. In VTAM, a set of entries for one or more logon modes. Each logon mode is identified by a logon mode name. Synonymous with *logmode table*.

logon-interpret routine. In VTAM, an installation exit routine, associated with an interpret table entry, that translates logon information. It may also verify the logon.

LPDA. Link Problem Determination Aid.

LU. Logical unit.

LU type. In SNA, the classification of an LU-LU session in terms of the specific subset of SNA protocols and options supported by the logical units (LUs) for that session, namely:

The mandatory and optional values allowed in the session activation request.

The usage of data stream controls, function management headers (FMHs), request unit (RU) parameters, and sense codes.

Presentation services protocols such as those associated with FMH usage.

LU types 0, 1, 2, 3, 4, 6.1, 6.2, and 7 are defined.

LU 6.2. Logical unit 6.2.

LU-LU session. In SNA, a session between two logical units (LUs) in an SNA network. It provides communication between two end users, or between an end user and an LU services component.

LU-LU session type. A deprecated term for *LU type*.

M

macroinstruction. (1) An instruction that when executed causes the execution of a predefined sequence of instructions in the same source language. (2) In assembler programming, an assembler language statement that causes the assembler to process a predefined set of statements called a macro definition. The statements normally produced from the macro definition replace the macroinstruction in the program. See also *definition statement*.

maintenance services. In SNA, one of the types of network services in system services control points (SSCPs) and physical units (PUs). Maintenance services provide facilities for testing links and nodes and for collecting and recording error information. See also *configuration services, management services, network services*, and *session services*.

major node. In VTAM, a set of resources that can be activated and deactivated as a group. See *node* and *minor node*.

management services. In SNA, one of the types of network services in control points (CPs) and physical units (PUs). Management services are the services provided to assist in the management of SNA networks, such as problem management, performance and accounting management, configuration management and change management. See also *configuration services, maintenance services, network services*, and *session services*.

mapped conversation. A type of conversation in which the data to be sent or received can be in a user-defined format. A logical unit (LU) that supports mapped conversations converts the user data to a format suitable for the basic conversation protocol boundary. See also *conversation* and *basic conversation*.

message. (1) (TC97) A group of characters and control bit sequences transferred as an entity. (2) In VTAM, the amount of function management data (FMD) transferred to VTAM by the application program with one SEND request.

message switching. (1) * (ISO) In a data network, the process of routing messages by receiving, storing, and forwarding complete messages. (2) The technique of receiving a complete message, storing, and then forwarding it unaltered to its destination.

migration. Installing a new version or release of a program when an earlier version or release is already in place.

minidisk. Synonym for *virtual disk*.

minor node. In VTAM, a uniquely-defined resource within a major node. See *node* and *major node*.

modem. A device that modulates and demodulates signals transmitted over data communication facilities. The term is a contraction for modulator-demodulator.

module. * A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading; for example, the input to or output from an assembler, compiler, linkage editor, or executive routine.

monitor. In the IBM Token-Ring Network, the function required to initiate the transmission of a token on the ring and to provide soft-error recovery in case of lost tokens, circulating frames, or other difficulties. The capability is present in all ring stations.

Multiple Virtual Storage (MVS). An IBM licensed program whose full name is the Operating System/Virtual Storage (OS/VS) with Multiple Virtual Storage/System Product for System/370. It is a software operating system controlling the execution of programs.

Multiple Virtual Storage for Extended Architecture (MVS/XA). An IBM licensed program whose full name is the Operating System/Virtual Storage (OS/VS) with Multiple Virtual Storage/System Product for Extended Architecture. Extended architecture allows 31-bit storage addressing. MVS/XA is a software operating system controlling the execution of programs.

multiple-domain network. In SNA, a network with more than one system services control point (SSCP). Contrast with *single-domain network*.

multitasking. The ability of an operating system to run several programs concurrently.

MVS. Multiple Virtual Storage.

MVS/ESA. Multiple Virtual Storage/Enterprise Systems Architecture.

MVS/XA. Multiple Virtual Storage for Extended Architecture.

N

NAU. Network addressable unit.

NCCF. Network Communications Control Facility.

NCP. (1) Network Control Program (IBM licensed program). Its full name is Advanced Communications Function for the Network Control Program. Synonymous with *ACF/NCP*. (2) Network control program (general term).

negative response (NR). In SNA, a response indicating that a request did not arrive successfully or was not processed successfully by the receiver. Contrast with *positive response*. See *exception response*.

NetView. A system 370-based IBM licensed program used to monitor a network, manage it, and diagnose its problems.

NetView command list language. An interpretive language unique to the NetView program that is used to write command lists.

NetView help desk.. In the NetView program, an online information facility that guides the help desk operator through problem management procedures.

NetView-NetView task (NNT). The task under which a cross-domain NetView operator session runs. See *operator station task*.

NetView/PC. A PC-based IBM licensed program through which application programs can be used to monitor, manage, and diagnose problems in IBM Token-Ring networks, non-SNA communication devices, and voice networks.

network. (1) (TC97) An interconnected group of nodes. (2) In data processing, a user application network. See *path control network, public network, SNA network*, and *user-application network*.

network address. In SNA, an address, consisting of subarea and element fields, that identifies a link, a link station, or a network addressable unit. Subarea nodes use network addresses; peripheral nodes use local addresses. The boundary function in the subarea node to which a peripheral node is attached transforms local addresses to network addresses and vice versa. See *local address*. See also *network name*.

network addressable unit (NAU). In SNA, a logical unit, a physical unit, or a system services control point. It is the origin or the destination of information transmitted by the path control network. Each NAU has a network address that represents it to the path control network. See also *network name, network address*, and *path control network*.

Network Communications Control Facility (NCCF). An IBM licensed program that is a base for command processors that can monitor, control, automate, and improve the operations of a network. Its function is included and enhanced in NetView's command facility.

network control program. A program, generated by the user from a library of IBM-supplied modules, that controls the operation of a communication controller.

Network Control Program (NCP). An IBM licensed program that provides communication controller support for single-domain, multiple-domain, and interconnected network capability. Its full name is Advanced Communications Function for the Network Control Program.

network log. A file that contains all messages processed by the NetView program.

network management vector transport (NMVT). A management services request/response unit (RU) that flows over an active session between physical unit management services and control point management services (SSCP-PU session).

network name. (1) In SNA, the symbolic identifier by which end users refer to a network addressable unit (NAU), a link, or a link station. See also *network address*. (2) In a multiple-domain network, the name of the APPL statement defining a VTAM application program is its network name and it must be unique across domains. Contrast with *ACB name*. See *uninterpreted name*.

network operator. (1) A person or program responsible for controlling the operation of all or part of a network. (2) The person or program that controls all the domains in a multiple-domain network. Contrast with *domain operator*.

Network Problem Determination Application (NPDA). An IBM licensed program that helps you identify network problems, such as hardware, software, and microcode, from a central control point using interactive display techniques. It runs as an NCCF communication network management (CNM) application program. Its function is included and enhanced in NetView's hardware monitor.

network services (NS). In SNA, the services within network addressable units (NAUs) that control network operation through SSCP-SSCP, SSCP-PU, and SSCP-LU sessions. See *configuration services, maintenance services, management services*, and *session services*.

NMVT. Network management vector transport.

NNT. NetView-NetView task.

no response. In SNA, a value in the form-of-response-requested field of the request header (RH) indicating that no response is to be returned to the request, whether or not the request is received and processed successfully. Contrast with *definite response* and *exception response*.

node. (1) In SNA, an endpoint of a link or junction common to two or more links in a network. Nodes can be distributed to host processors, communication controllers, cluster controllers, or terminals. Nodes can vary in routing and other functional capabilities. See *boundary node*, *host node*, *peripheral node*, and *subarea node*. (2) In VTAM, a point in a network defined by a symbolic name. See *major node* and *minor node*.

node name. In VTAM, the symbolic name assigned to a specific major or minor node during network definition.

node type. In SNA, a designation of a node according to the protocols it supports and the network addressable units (NAUs) that it can contain. Five types are defined: 1, 2.0, 2.1, 4, and 5. Type 1, type 2.0, and type 2.1 nodes are peripheral nodes; type 4 and type 5 nodes are subarea nodes. See also *type 2.1 node*.

NPDA. Network Problem Determination Application.

O

online. Stored in a computer and accessible from a terminal.

operand. (1) (ISO) An entity on which an operation is performed. (2) * That which is operated upon. An operand is usually identified by an address part of an instruction. (3) Information entered with a command name to define the data on which a command processor operates and to control the execution of the command processor. (4) An expression to whose value an operator is applied. See also *definition statement*, *keyword*, *keyword parameter*, and *parameter*.

operator. (1) In a language statement, the lexical entity that indicates the action to be performed on operands. (2) A person who operates a machine. See *network operator*. See also *definition statement*.

operator station task (OST). The NetView task that establishes and maintains the online session with the network operator. There is one operator station task for each network operator who logs on to the NetView program. See *NetView-NetView task*.

OST. Operator station task.

P

pacing. In SNA, a technique by which a receiving component controls the rate of transmission of a sending component to prevent overrun or congestion. See *session-level pacing, send pacing*, and *virtual route (VR) pacing*. See also *flow control*.

pacing response. In SNA, an indicator that signifies a receiving component's readiness to accept another pacing group; the indicator is carried in a response header (RH) for session-level pacing, and in a transmission header (TH) for virtual route pacing.

packet mode operation. Synonym for *packet switching*.

packet switching. (1) (ISO) The process of routing and transferring data by means of addressed packets so that a channel is occupied only during the transmission of a packet. On completion of the transmission, the channel is made available for the transfer of other packets. (2) Synonymous with *packet mode operation*. See also *circuit switching*.

page. (1) The portion of a panel that is shown on a display surface at one time. (2) To move back and forth among the pages of a multiple-page panel. See also *scroll*. (3) (ISO) In a virtual storage system, a fixed-length block that has a virtual address and that can be transferred between real storage and auxiliary storage. (4) To transfer instructions, data, or both between real storage and external page or auxiliary storage.

panel. (1) A formatted display of information that appears on a terminal screen. See also *help panel* and *task panel*. Contrast with *screen*. (2) In computer graphics, a display image that defines the locations and characteristics of display fields on a display surface.

parameter. (1) (ISO) A variable that is given a constant value for a specified application and that may denote the application. (2) An item in a menu for which the user specifies a value or for which the system provides a value when the menu is interpreted. (3) Data passed to a program or procedure by a user or another program, namely as an operand in a language statement, as an item in a menu, or as a shared data structure. See also *keyword, keyword parameter*, and *operand*.

partitioned data set (PDS). A data set in direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

path. (1) In SNA, the series of path control network components (path control and data link control) that are traversed by the information exchanged between two network addressable units (NAUs). See also *explicit route (ER), route extension*, and *virtual route (VR)*. (2) In VTAM when defining a switched major node, a potential dial-out port that can be used to reach that node. (3) In the NetView/PC program, a complete line in a configuration that contains all of the resources in the service point command service (SPCS) query link configuration request list.

path control (PC). The function that routes message units between network addressable units (NAUs) in the network and provides the paths between them. It converts the BIUs from transmission control (possibly segmenting them) into path information units (PIUs) and exchanges basic transmission units (BTUs) and one or more PIUs with data link control. Path control differs for peripheral nodes, which use local addresses for routing, and subarea nodes, which use network addresses for routing. See *peripheral path control* and *subarea path control*. See also *link, peripheral node*, and *subarea node*.

path control (PC) layer. In SNA, the layer that manages the sharing of link resources of the SNA network and routes basic information units (BIUs) through it. See also *BIU segment, blocking of PIUs, data link control layer*, and *transmission control layer*.

path control (PC) network. In SNA, the part of the SNA network that includes the data link control and path control layers. See *SNA network* and *user application network*. See also *boundary function*.

path information unit (PIU). In SNA, a message unit consisting of a transmission header (TH) alone, or of a TH followed by a basic information unit (BIU) or a BIU segment. See also *transmission header*.

PC. (1) Path control. (2) Personal Computer. Its full name is the IBM Personal Computer.

PCID. Procedure-correlation identifier.

performance class. In the NetView program, a description of an objective or commitment of performance. It consists of a performance class name, boundary definitions, response time definition, response time ranges, and response time percentage objectives. Sessions may be assigned performance classes.

peripheral host node. A node that provides an application program interface (API) for running application programs but does not provide SSCP functions and is not aware of the network configuration. The peripheral host node does not provide subarea node services. It has boundary function provided by its adjacent subarea. See *boundary node, host node, node, peripheral node, subarea host node*, and *subarea node*. See also *boundary function* and *node type*.

peripheral LU. In SNA, a logical unit representing a peripheral node.

peripheral node. In SNA, a node that uses local addresses for routing and therefore is not affected by changes in network addresses. A peripheral node requires boundary-function assistance from an adjacent subarea node. A peripheral node is a physical unit (PU) type 1, 2.0, or 2.1 node connected to a subarea node with boundary function within a subarea. See *boundary node, host node, node, peripheral host node, subarea host node*, and *subarea node*. See also *boundary function* and *node type*.

peripheral path control. The function in a peripheral node that routes message units between units with local addresses and provides the paths between them. See *path control* and *subarea path control*. See also *boundary function, peripheral node*, and *subarea node*.

peripheral PU. In SNA, a physical unit representing a peripheral node.

Personal Computer (PC). The IBM Personal Computer line of products including the 5150 and subsequent models.

physical connection. In VTAM, a point-to-point connection or multipoint connection. Synonymous with *connection*.

physical unit (PU). In SNA, a type of network addressable unit (NAU). A physical unit (PU) manages and monitors the resources (such as attached links) of a node, as requested by a system services control point (SSCP) through an SSCP-PU session. An SSCP activates a session with the physical unit in order to indirectly manage, through the PU, resources of the node such as attached links. See also *peripheral PU* and *subarea PU*.

physical unit (PU) services. In SNA, the components within a physical unit (PU) that provide configuration services and maintenance services for SSCP-PU sessions. See also *logical unit (LU) services*.

PLU. Primary logical unit.

POI. Programmed operator interface.

positional operand. An operand in a language statement that has a fixed position. See also *definition statement*. Contrast with *keyword operand*.

positive response. A response indicating that a request was received and processed. Contrast with *negative response*.

POST. Power-on self test. A series of diagnostic tests that are run each time the computer's power is turned on.

PPT. Primary POI task.

primary half-session. In SNA, the half-session that sends the session activation request. See also *primary logical unit*. Contrast with *secondary half-session*.

primary logical unit (PLU). In SNA, the logical unit (LU) that contains the primary half-session for a particular LU-LU session. Each session must have a PLU and secondary logical unit (SLU). The PLU is the unit responsible for the bind and is the controlling LU for the session. A particular LU may contain both primary and secondary half-sessions for different active LU-LU sessions. Contrast with *secondary logical unit (SLU)*.

primary POI task (PPT). The NetView subtask that processes all unsolicited messages received from the VTAM program operator interface (POI) and delivers them to the controlling operator or to the command processor. The PPT also processes the initial command specified to execute when the NetView program is initialized and timer request commands scheduled to execute under the PPT.

problem determination. The process of identifying the source of a problem; for example, a program component, a machine failure, telecommunication facilities, user or contractor-installed programs or equipment, an environment failure such as a power loss, or a user error.

procedure-correlation identifier (PCID). In SNA, a value used by a control point to correlate requests and replies.

product-set identification (PSID). (1) In SNA, a technique for identifying the hardware and software products that implement a network component. (2) A management services common subvector that transports the information described in definition (1).

program temporary fix (PTF). A temporary solution or bypass of a problem diagnosed by IBM in a current unaltered release of the program.

program-to-program interface. In the NetView program, a facility that allows user programs to send data buffers to or receive data buffers from other user programs. It also allows system and application programs to send alerts to the NetView hardware monitor.

programmed operator. A VTAM application program that is authorized to issue VTAM operator commands and receive VTAM operator awareness messages. See also *solicited messages* and *unsolicited messages*.

programmed operator interface (POI). A VTAM function that allows programs to perform VTAM operator functions.

PSID. Product-set identification.

PTF. Program temporary fix.

PU. Physical unit.

PU-PU flow. In SNA, the exchange between physical units (PUs) of network control requests and responses.

public network. A network established and operated by communication common carriers or telecommunication Administrations for the specific purpose of providing circuit-switched, packet switched, and leased-circuit services to the public. Contrast with *user-application network*.

R

real name. The name by which a logical unit (LU), logon mode table, or class-of-service (COS) table is known within the SNA network in which it resides.

receive pacing. In SNA, the pacing of message units that the component is receiving. See also *send pacing*.

Recommendation X.21 (Geneva 1980). A Consultative Committee on International Telegraph and Telephone (CCITT) recommendation for a general purpose interface between data terminal equipment and data circuit equipment for synchronous operations on a public data network.

Recommendation X.25 (Geneva 1980). A Consultative Committee on International Telegraph and Telephone (CCITT) recommendation for the interface between data terminal equipment and packet-switched data networks. See also *packet switching.*

recommended action. Procedures suggested by the NetView program that can be used to determine the causes of network problems.

record. (1) (ISO) In programming languages, an aggregate that consists of data objects, possibly with different attributes, that usually have identifiers attached to them. In some programming languages, records are called structures. (2) (TC97) A set of data treated as a unit. (3) A set of one or more related data items grouped for processing. (4) In VTAM, the unit of data transmission for record mode. A record represents whatever amount of data the transmitting node chooses to send.

recording filter. In the NetView program, the function that determines which events, statistics, and alerts are stored on a data base.

regular command. In the NetView program, any VTAM or NetView command that is not an immediate command and is processed by a regular command processor. Contrast with *immediate command.*

release. For VTAM, to relinquish control of resources (communication controllers or physical units). See also *resource takeover.* Contrast with *acquire (2).*

remote. Concerning the peripheral parts of a network not centrally linked to the host processor and generally using telecommunication lines with public right-of-way.

request header (RH). In SNA, control information preceding a request unit (RU). See also *request/response header (RH).*

request/response header (RH). In SNA, control information, preceding a request/response unit (RU), that specifies the type of RU (request unit or response unit) and contains control information associated with that RU.

reset. On a virtual circuit, reinitialization of data flow control. At reset, all data in transit are eliminated.

resource. (1) Any facility of the computing system or operating system required by a job or task, and including main storage, input/output devices, the processing unit, data sets, and control or processing programs. (2) In the NetView program, any hardware or software that provides function to the network.

resource hierarchy. In VTAM, the relationship among network resources in which some resources are subordinate to others as a result of their position in the network structure and architecture; for example, the logical units (LUs) of a peripheral physical unit (PU) are subordinate to that PU, which, in turn, is subordinate to the link attaching it to its subarea node.

resource takeover. In VTAM, action initiated by a network operator to transfer control of resources from one domain to another. See also *acquire (2)* and *release.* See *takeover.*

resource types. In the NetView program, a concept to describe the organization of panels. Resource types are defined as central processing unit, channel, control unit, and I/O device for one category; and communication controller, adapter, link, cluster controller, and terminal for another category. Resource types are combined with data types and display types to describe display organization. See also *data types* and *display types.*

response. A reply represented in the control field of a response frame. It advises the primary or combined station of the action taken by the secondary or other combined station to one or more commands. See also *command.*

response header (RH). In SNA, a header, optionally followed by a response unit (RU), that indicates whether the response is positive or negative and that may contain a pacing response. See also *negative response, pacing response*, and *positive response.*

response time. (1) The amount of time it takes after a user presses the enter key at the terminal until the reply appears at the terminal. (2) For response time monitoring, the time from the activation of a transaction until a response is received, according to the response time definition coded in the performance class.

response time monitor (RTM). A feature available with certain hardware devices to allow measurement of response times, which may be collected and displayed by the NetView program.

Restructured Extended Executor (REXX). An interpretive language used to write command lists.

return code. * A code [returned from a program] used to influence the execution of succeeding instructions.

REXX. Restructured Extended Executor.

RH. Request/response header.

ring. A network configuration where a series of attaching devices are connected by unidirectional transmission links to form a closed path.

route. See *explicit route* and *virtual route.*

route extension (REX). In SNA, the path control network components, including a peripheral link, that make up the portion of a path between a subarea node and a network addressable unit (NAU) in an adjacent peripheral node. See also *path, explicit route (ER)* and *virtual route (VR).*

RPL exit routine. In VTAM, an application program exit routine whose address has been placed in the EXIT field of a request parameter list (RPL). VTAM invokes the routine to indicate that an asynchronous request has been completed. See *EXLST exit routine*.

RTM. Response time monitor.

S

scanner. (1) A device capable of electronically reviewing amounts of data and translating the data into a machine readable form. (2) For the 3725 communication controller, a processor dedicated to controlling a small number of telecommunication lines. It provides the connection between the line interface coupler hardware and the central control unit.

scope of commands. In the NetView program, the facility that provides the ability to assign different responsibilities to various operators.

screen. An illuminated display surface; for example, the display surface of a CRT or plasma panel. Contrast with *panel*.

scroll. To move all or part of the display image vertically to display data that cannot be observed within a single display image. See also *page (2)*.

SDLC. Synchronous Data Link Control.

secondary half-session. In SNA, the half-session that receives the session-activation request. See also *secondary logical unit (SLU)*. Contrast with *primary half-session*.

secondary logical unit (SLU). In SNA, the logical unit (LU) that contains the secondary half-session for a particular LU-LU session. An LU may contain secondary and primary half-sessions for different active LU-LU sessions. Contrast with *primary logical unit (PLU)*.

secondary logical unit (SLU) key. A key-encrypting key used to protect a session cryptography key during its transmission to the secondary half-session.

segment. (1) In the IBM Token-Ring Network, a section of cable between components or devices on the network. A segment may consist of a single patch cable, multiple patch cables connected together, or a combination of building cable and patch cables connected together. (2) See *link connection segment*.

send pacing. In SNA, pacing of message units that a component is sending. See also *receive pacing*.

Service Level Reporter (SLR). A licensed program that generates management reports from data sets such as System Management Facility (SMF) files.

service point (SP). An entry point that supports applications that provide network management for resources not under the direct control of itself as an entry point. Each resource is either under the direct control of another entry point or not under the direct control of any entry point. A service point accessing these resources is not required to use SNA sessions (unlike a focal point). A service point is needed when entry point support is not yet available for some network management function.

service point command service (SPCS). An extension of the command facility in the NetView program that allows the host processor to communicate with a service point by using the communication network management (CNM) interface.

session. In SNA, a logical connection between two network addressable units (NAUs) that can be activated, tailored to provide various protocols, and deactivated, as requested. Each session is uniquely identified in a transmission header (TH) by a pair of network addresses, identifying the origin and destination NAUs of any transmissions exchanged during the session. See *half-session*, *LU-LU session*, *SSCP-LU session*, *SSCP-PU session*, and *SSCP-SSCP session*. See also *LU-LU session type* and *PU-PU flow*.

session awareness (SAW) data. Data collected by the NetView program about a session that includes the session type, the names of session partners, and information about the session activation status. It is collected for LU-LU, SSCP-LU, SSCP-PU, and SSCP-SSCP sessions and for non-SNA terminals not supported by NTO. It can be displayed in various forms, such as most recent sessions lists.

session data. Data about a session, collected by the NetView program, that consists of session awareness data, session trace data, and session response time data.

session monitor. The component of the NetView program that collects and correlates session-related data and provides online access to this information.

session parameters. In SNA, the parameters that specify or constrain the protocols (such as bracket protocol and pacing) for a session between two network addressable units. See also *logon mode*.

session services. In SNA, one of the types of network services in the control point (CP) and in the logical unit (LU). These services provide facilities for an LU or a network operator to request that the SSCP initiate or terminate sessions between logical units. See *configuration services*, *maintenance services*, and *management services*.

session-initiation request. In SNA, an Initiate or logon request from a logical unit (LU) to a control point (CP) that an LU-LU session be activated.

session-level pacing. In SNA, a flow control technique that permits a receiver to control the data transfer rate (the rate at which it receives request units) on the normal flow. It is used to prevent overloading a receiver with unprocessed requests when the sender can generate requests faster than the receiver can process them. See also *pacing* and *virtual route pacing*.

simulated logon. A session-initiation request generated when a VTAM application program issues a SIMLOGON macroinstruction. The request specifies a logical unit (LU) with which the application program wants a session in which the requesting application program will act as the primary logical unit (PLU).

single-domain network. In SNA, a network with one system services control point (SSCP). Contrast with *multiple-domain network*.

SLR. Service Level Reporter.

SLU. Secondary logical unit.

SMF. System management facility.

SNA. Systems Network Architecture.

SNA network. The part of a user-application network that conforms to the formats and protocols of Systems Network Architecture. It enables reliable transfer of data among end users and provides protocols for controlling the resources of various network configurations. The SNA network consists of network addressable units (NAUs), boundary function components, and the path control network.

SNBU. Switched network backup.

SP. Service point.

span. In the NetView program, a user-defined group of network resources within a single domain. Each major or minor node is defined as belonging to one or more spans. See also *span of control*.

span of control. The total network resources over which a particular network operator has control. All the network resources listed in spans associated through profile definition with a particular network operator are within that operator's span of control.

SPCS. Service point command service.

SS. Start-stop.

SSCP. System services control point.

SSCP-LU session. In SNA, a session between a system services control point (SSCP) and a logical unit (LU); the session enables the LU to request the SSCP to help initiate LU-LU sessions.

SSCP-PU session. In SNA, a session between a system services control point (SSCP) and a physical unit (PU); SSCP-PU sessions allow SSCPs to send requests to and receive status information from individual nodes in order to control the network configuration.

SSCP-SSCP session. In SNA, a session between the system services control point (SSCP) in one domain and the SSCP in another domain. An SSCP-SSCP session is used to initiate and terminate cross-domain LU-LU sessions.

SSP. System Support Programs (IBM licensed program). Its full name is Advanced Communications Function for System Support Programs. Synonymous with *ACF/SSP*.

ST. Session configuration screen abbreviation.

statement. A language syntactic unit consisting of an operator, or other statement identifier, followed by one or more operands. See *definition statement*.

station. (1) One of the input or output points of a network that uses communication facilities; for example, the telephone set in the telephone system or the point where the business machine interfaces with the channel on a leased private line. (2) One or more computers, terminals, or devices at a particular location.

subarea. A portion of the SNA network consisting of a subarea node, any attached peripheral nodes, and their associated resources. Within a subarea node, all network addressable units, links, and adjacent link stations (in attached peripheral or subarea nodes) that are addressable within the subarea share a common subarea address and have distinct element addresses.

subarea host node. A host node that provides both subarea function and an application program interface (API) for running application programs. It provides system services control point (SSCP) functions, subarea node services, and is aware of the network configuration. See *boundary node*, *communication management configuration host node*, *data host node*, *host node*, *node*, *peripheral node*, and *subarea node*. See also *boundary function* and *node type*.

subarea link. In SNA, a link that connects two subarea nodes. See *channel link* and *link*.

subarea node. In SNA, a node that uses network addresses for routing and whose routing tables are therefore affected by changes in the configuration of the network. Subarea nodes can provide gateway function, and boundary function support for peripheral nodes. Type 4 and type 5 nodes are subarea nodes. See *boundary node*, *host node*, *node*, *peripheral node*, and *subarea host node*. See also *boundary function* and *node type*.

subarea path control. The function in a subarea node that routes message units between network addressable units (NAUs) and provides the paths between them. See *path control* and *peripheral path control*. See also *boundary function*, *peripheral node*, and *subarea node*.

subarea PU. In SNA, a physical unit (PU) in a subarea node.

subsystem. A secondary or subordinate system, usually capable of operating independent of, or asynchronously with, a controlling system.

subvector. A subcomponent of the MAC major vector.

supervisor. The part of a control program that coordinates the use of resources and maintains the flow of processing unit operations.

suppression character. In the NetView program, a user-defined character that is coded at the beginning of a command list statement or a command to prevent the statement or command from appearing on the operator's terminal screen or in the network log.

switched network. Any network in which connections are established by closing switches, for example, by dialing.

switched network backup (SNBU). An optional facility that allows a user to specify, for certain types of PUs, a switched line to be used as an alternate path if the primary line becomes unavailable or unusable.

Synchronous Data Link Control (SDLC). A discipline for managing synchronous, code-transparent, serial-by-bit information transfer over a link connection. Transmission exchanges may be duplex or half-duplex over switched or nonswitched links. The configuration of the link connection may be point-to-point, multipoint, or loop. SDLC conforms to subsets of the Advanced Data Communication Control Procedures (ADCCP) of the American National Standards Institute and High-Level Data Link Control (HDLC) of the International Standards Organization.

system management facility (SMF). A standard feature of MVS that collects and records a variety of system and job-related information.

system services control point (SSCP). In SNA, a central location point within an SNA network for managing the configuration, coordinating network operator and problem determination requests, and providing directory support and other session services for end users of the network. Multiple SSCPs, cooperating as peers, can divide the network into domains of control, with each SSCP having a hierarchical control relationship to the physical units and logical units within its domain.

system services control point (SSCP) domain. The system services control point and the physical units (PUs), logical units (LUs), links, link stations and all the resources that the SSCP has the ability to control by means of activation requests and deactivation requests.

System Support Programs (SSP). An IBM licensed program, made up of a collection of utilities and small programs, that supports the operation of the NCP.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through and controlling the configuration and operation of networks.

T

takeover. The process by which the failing active subsystem is released from its extended recovery facility (XRF) sessions with terminal users and replaced by an alternate subsystem. See *resource takeover*.

task. A basic unit of work to be accomplished by a computer. The task is usually specified to a control program in a multiprogramming or multiprocessing environment.

task panel. Online display from which you communicate with the program in order to accomplish the program's function, either by selecting an option provided on the panel or by entering an explicit command. See *help panel*.

telecommunication line. Any physical medium such as a wire or microwave beam, that is used to transmit data. Synonymous with *transmission line*.

terminal. A device that is capable of sending and receiving information over a link; it is usually equipped with a keyboard and some kind of display, such as a screen or a printer.

TH. Transmission header.

threshold. In the NetView program, refers to a percentage value set for a resource and compared to a calculated error-to-traffic ratio.

threshold analysis and remote access. (1) A component of the NetView program that can notify a central operator about network problems and errors. It provides remote control of IBM 3600 and 4700 controllers and can record, analyze, and display performance and status data on IBM 3600 and 4700 Finance Communications Systems. (2) The feature of the back-level

NPDA licensed program that performs some of these functions.

time sharing option (TSO). An optional configuration of the operating system that provides conversational time sharing from remote stations.

token. A sequence of bits passed from one device to another along the token ring. When the token has data appended to it, it becomes a frame.

token ring. A network with a ring topology that passes tokens from one attaching device to another. For example, the IBM Token-Ring Network.

transaction program. (1) A program that is executed by or within an application program and performs services related to the processing of a transaction. (2) In VTAM, a program that performs services related to the processing of a transaction. One or more transaction programs may operate within a VTAM application program that is using the VTAM application program interface (API). In that situation, the transaction program would request services from the application program, using protocols defined by that application program. The application program, in turn, could request services from VTAM by issuing the APPCCMD macroinstruction.

transmission control (TC) layer. In SNA, the layer within a half-session that synchronizes and paces session-level data traffic, checks session sequence numbers of requests, and enciphers and deciphers end-user data. Transmission control has two components: the connection point manager and session control. See also *half-session.*

transmission header (TH). In SNA, control information, optionally followed by a basic information unit (BIU) or a BIU segment, that is created and used by path control to route message units and to control their flow within the network. See also *path information unit.*

transmission line. Synonym for *telecommunication line.*

TSO. Time sharing option.

type 2.1 node (T2.1 node). A node that can attach to an SNA network as a peripheral node using the same protocols as type 2.0 nodes. Type 2.1 nodes can be directly attached to one another using peer-to-peer protocols. See *end node*, *node*, and *subarea node.* See also *node type.*

type 2.1 node (T2.1 node) control point domain. The CP, its logical units (LUs), links, link stations, and all resources that it activates and deactivates.

U

uninterpreted name. In SNA, a character string that a system services control point (SSCP) is able to convert into the network name of a logical unit (LU). Typically, an uninterpreted name is used in a logon or Initiate request from a secondary logical unit (SLU) to identify the primary logical unit (PLU) with which the session is requested.

upstream. In the direction of data flow from the end user to the host. Contrast with *downstream.*

user. Anyone who requires the services of a computing system.

user exit. A point in an IBM-supplied program at which a user routine may be given control.

user exit routine. A user-written routine that receives control at predefined user exit points. User exit routines can be written in assembler or a high-level language (HLL).

user table. In TPNS, one or more text data entries contained in a table format, which may be referred to for logic testing and message generation.

user-application network. A configuration of data processing products, such as processors, controllers, and terminals, established and operated by users for the purpose of data processing or information exchange, which may use services offered by communication common carriers or telecommunication Administrations. Contrast with *public network.*

using node. (1) In NCP, the NCP in the host's domain that reports a link error condition. (2) For the command facility of the NetView program and for NCCF, the ID parameter of certain network control commands.

V

value. (1) (TC97) A specific occurrence of an attribute, for example, "blue" for the attribute "color." (2) A quantity assigned to a constant, a variable, a parameter, or a symbol.

variable. In the NetView program, a character string beginning with & that is coded in a command list and is assigned a value during execution of the command list.

vector. The MAC frame information field.

viewing filter. In the NetView program, the function that allows a user to select the data to be displayed on a terminal. All other stored data is blocked.

virtual disk. (1) A logical subdivision (or all) of a physical disk pack in the VM operating system that has its own virtual device address, consecutive virtual cylinders, and a volume table of contents (VTOC) or disk label identifier. (2) Synonymous with *minidisk*.

virtual machine. A functional simulation of a computer and its associated devices.

Virtual Machine (VM). ·A licensed program whose full name is the Virtual Machine/System Product (VM/SP). It is a software operating system that manages the resources of a real processor to provide virtual machines to end users. As a time-sharing system control program, it consists of the virtual machine control program (CP), the conversational monitor system (CMS), the group control system (GCS), and the interactive problem control system (IPCS).

Virtual Machine/System Product High Performance Option. An IBM licensed program that can be installed and executed in conjunction with VM/System Product to extend the capabilities of the VM/System Product with programming enhancements, support for microcode assists, and additional functions. The VM/SP High Performance Option program package is not executable by itself. It requires installation of VM/System Product or an equivalent IBM-licensed program.

virtual route (VR). In SNA, a logical connection (1) between two subarea nodes that is physically realized as a particular explicit route, or (2) that is contained wholly within a subarea node for intranode sessions. A virtual route between distinct subarea nodes imposes a transmission priority on the underlying explicit route, provides flow control through virtual-route pacing, and provides data integrity through sequence numbering of path information units (PIUs). See also *explicit route (ER), path*, and *route extension*.

virtual route (VR) pacing. In SNA, a flow control technique used by the virtual route control component of path control at each end of a virtual route to control the rate at which path information units (PIUs) flow over the virtual route. VR pacing can be adjusted according to traffic congestion in any of the nodes along the route. See also *pacing* and *session-level pacing*.

virtual route selection exit routine. In VTAM, an optional installation exit routine that modifies the list of virtual routes associated with a particular class of service before a route is selected for a requested LU-LU session.

Virtual Storage Access Method (VSAM). An access method for direct or sequential processing of fixed and variable-length records on direct access devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry-sequence), or by relative-record number.

Virtual Storage Extended (VSE). An IBM licensed program whose full name is the Virtual Storage Extended/Advanced Function. It is a software operating system controlling the execution of programs.

Virtual Telecommunications Access Method (VTAM). An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability.

VM. Virtual Machine. Its full name is Virtual Machine/System Product. Synonymous with *VM/SP*.

VM/SP. Virtual Machine/System Product. Synonym for *VM*.

VM/XA. Virtual Machine/Extended Architecture.

VR. Virtual route.

VSAM. Virtual Storage Access Method.

VSE. Virtual Storage Extended. Synonymous with *VSE/AF*.

VSE/AF. Virtual Storage Extended/Advanced Function. Synonym for *VSE*.

VTAM. Virtual Telecommunications Access Method (IBM licensed program). Its full name is Advanced Communications Function for the Virtual Telecommunications Access Method. Synonymous with *ACF/VTAM*.

VTAM operator command. A command used to monitor or control a VTAM domain. See also *definition statement*.

W

wide area network. A network that provides data communication capability in geographic areas larger than those serviced by local area networks. Wide area networks may extend across public rights-of-way. Contrast with *local area network*.

wrap. In general, to go from the maximum to the minimum in computer storage. For example, the continuation of an operation from the maximum value in storage to the first minimal value.

wrap count. In the NetView program, the number of events that can be retained on the data base for a specific resource.

X

X.25. See *Recommendation X.25 (Geneva 1980).*

XA. See *extended architecture.*

Bibliography

NetView Publications

Learning About NetView: Operator Training (SK2T-0292) is an interactive PC-based operator training package that teaches SNA and basic network management concepts to new and inexperienced NetView operators. This training package uses graphics, animation and interactive NetView product simulations in a series of lessons to teach the basics of NetView operation.

NetView Installation and Administration Guide (SC31-6018) helps system programmers install and prepare the NetView program for operation. It is arranged in a simplified, step-by-step style and is meant to be used in conjunction with the sample network documented in *Network Program Products Samples*.

NetView Administration Reference (SC31-6014) is for system programmers and network operators who need a complete understanding of the NetView resource definition statements. This book lists each statement in alphabetical order giving its purpose and location.

NetView Tuning Guide (SC31-6079)[2] describes methods for controlling and improving the performance of the NetView Release 3 program. It is designed for system programmers who need to understand how NetView tuning values are determined and optimized.

NetView Customization Guide (SC31-6016) is designed for system programmers and others who want to customize the NetView program to reflect their network's needs or operating procedures. This book focuses on the different application programming interfaces that can be customized and explains how to modify NetView help panels and problem determination displays.

NetView Customization: Using PL/I and C (SC31-6037) describes the ways system programmers can tailor the NetView program to satisfy unique requirements or operating procedures. It discusses the uses and advantages of user-written programs (exit routines, command processors, and subtasks). It also provides instructions in designing, writing, and installing user-written programs in PL/I and C.

NetView Customization: Using Assembler (SC31-6078) describes the ways system programmers can tailor the NetView program to satisfy unique requirements or operating procedures. It discusses the uses and advantages of user-written programs (exit routines, command processors, and subtasks). It also provides instructions in designing, writing, and installing user-written programs in assembler.

NetView Customization: Writing Command Lists (SC31-6015) explains how to simplify network operator tasks by using command lists. It provides step-by-step instructions for writing simple and advanced command lists and for migrating from NCCF message automation to NetView message automation.

NetView Operation Primer (SC31-6020) provides a basic description of the network management task for new network operators. Topics include starting and stopping a network, controlling resources, monitoring a network, and gathering the necessary data to report a problem.

NetView Operation (SC31-6019) provides system programmers and experienced network operators a comprehensive explanation of network management using the NetView program. Topics include detailed command explanation and panel flows, as well as information on how the various components interact with each other.

NetView Command Summary (SX75-0026) is a reference card that provides network operators with the format of all the commands and the commonly used NetView command lists. The commands are listed in alphabetical order by component.

NetView Problem Determination and Diagnosis (LY43-0001) aids system programmers in identifying a NetView problem, classifying it, and describing it to an IBM Support Center.

NetView Problem Determination Supplement for Management Services Major Vectors 0001 and 0025 (LD21-0023) describes major vectors 0001 and 0025 for system programmers and network operators involved in problem determination or diagnosis. The supplement may be used for the generic alert option and other problem determination tasks.

NetView Resource Alerts Reference (SC31-6024) lists the messages sent by NetView-supported hardware and software resources. It helps system programmers analyze the messages into their component parts: action codes, event types, message text, and qualifiers. The book is a reference for those who need more information than online help provides.

[2] When available.

NetView Storage Estimates (SK2T-1988) is an interactive PC-based tool that helps the user estimate storage requirements for NetView. This tool can be used for planning, installation, and tuning purposes. It is intended for network planners, system programmers, and IBM service personnel.

Console Automation Using NetView: Planning (SC31-6058) describes an approach to automate the way a system handles messages and responses to alerts. It includes information you should know before beginning such automation, as well as sample plans and proposals you might find useful in promoting your automation concept. This book includes planning information for MVS, VM, and VSE users of the NetView program.

NetView/PC Publications

NetView/PC Planning, Installation, and Customization (SC31-6002) provides planning, installation, and customization information on NetView/PC and explains the communication requirements upstream to the host and downstream to supported devices. Information relating to the required PC environment and host products that support NetView/PC is also provided. It also discusses topics that are of general interest when you are ordering your equipment.

NetView/PC Application Program Interface/Communications Services Reference (SC31-6004) is a reference for OS/2 programmers who use the API/CS and for system programmers who write command processors to run under NetView. The API/CS provides a means for vendor and other external applications to use the communication services of NetView/PC.

NetView/PC Operation (SC31-6003) describes how to operate the program and diagnose problems in NetView/PC.

NetView/PC Quick Reference (SX75-0016) describes all of the functions of the F-keys throughout the NetView/PC program.

Other Network Program Products Publications

For more information about the books listed in this section, see *Bibliography and Master Index for NetView, NCP, and VTAM*.

Network Program Products General Information (GC30-3350)

Network Program Products Planning (SC30-3351)

Network Program Products Samples (SC30-3352)

Bibliography and Master Index for NetView, NCP, and VTAM (GC31-6081)[3]

VTAM Publications

The following list shows the books for VTAM V3R2. For information about the books for VTAM V3R1, V3R1.1, or V3R1.2, see any VTAM V3R2 book or the *Network Program Products Bibliography and Master Index*.

VTAM Installation and Resource Definition (SC23-0111)

VTAM Customization (LY30-5614)

VTAM Directory of Programming Interfaces for Customers (GC31-6403)

VTAM Operation (SC23-0113)

VTAM Messages and Codes (SC23-0114)

VTAM Programming (SC23-0115)

VTAM Programming for LU 6.2 (SC30-3400)

VTAM Diagnosis Guide (LY30-5601)

VTAM Data Areas for MVS (LY30-5592)

VTAM Data Areas for VM (LY30-5593)

VTAM Data Areas for VSE (LY30-5594)

VTAM Reference Summary (LY30-5600)

NCP, SSP, and EP Publications

The following list shows the related books for NCP V4 and NCP V5.

NCP, SSP, and EP Generation and Loading Guide (SC30-3348)

NCP, SSP, and Related Products Directory of Programming Interfaces for Customers (GC31-6202)

NCP Migration Guide (SC30-3252 for NCP V4 and SC30-3440 for NCP V5)

NCP, SSP, and EP Resource Definition Guide (SC30-3349 for NCP V4 and SC30-3447 for NCP V5)

[3] When available.

NCP, SSP, and EP Resource Definition Reference (SC30-3254 for NCP V4 and SC30-3448 for NCP V5)

NCP and EP Reference Summary and Data Areas (LY30-5570 for NCP V4 and LY30-5603 for NCP V5)

NCP Customization Guide (LY30-5571 for NCP V4 LY30-5606 for NCP V5)

NCP Customization Reference (LY30-5612 for NCP V4 and LY30-5607 for NCP V5)

SSP Customization (LY43-0021)

NCP, SSP, and EP Messages and Codes (SC30-3169)

NCP, SSP, and EP Diagnosis Guide (LY30-5591)

NCP and EP Reference (LY30-5569 for NCP V4 and LY30-5605 for NCP V5)

Related Publications

MVS/XA Supervisor Services and Macros (GC28-1154)

MVS/Extended Architecture Data Administration: Utilities (GC26-4018)

MVS/Extended Architecture Job Control Language User's Guide (GC28-1351)

VM/SP System Programmer's Guide (SC19-6203)

CMS Command and Macro Reference (SC19-6209)

Systems Network Architecture Product Formats (LY43-0081)

Systems Network Architecture Technical Overview (GC30-3073)

IBM 3270 Information Display System Data Stream Programmer's Reference (GA23-0059)

IBM 3290 Information Panel Description and Reference (GA23-0021)

IBM 3600/4700 Threshold Analysis and Remote Access Feature Installation and Customization Guide (SC34-2041)

PL/I Programming Guide (SC26-4307)

PL/I Programming: Language Reference (SC26-4308)

PL/I Programming: Messages and Codes (SC26-4309)

PL/I Installation and Customization Under MVS (SC26-4311)

C/370 User's Guide (SC09-1264

SAA: Common Programming Interface C Reference (SC26-4353)

VM/SP System Product Interpreter User's Guide (SC24-5238)

VM/SP System Product Interpreter Reference (SC24-5239)

TSO/E REXX User's Guide (SC28-1882)

TSO/E REXX Reference (SC28-1883)

Index

Reader's Comments

NetView
Customization Guide
Release 3

Publication No. SC31-6016-2

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Be sure to print your name and address below if you would like a reply.

Name	Address
Company or Organization	
Phone No.	

Reader's Comments
SC31-6016-2

IBM®

Cut or
Along

Fold and Tape

Please do not staple

Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL   PERMIT NO. 40   ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Information Development
Department E15
PO BOX 12195
RESEARCH TRIANGLE PARK NORTH CAROLINA 27709-9990

Fold and Tape

Please do not staple

Fold and Tape

SC31-6016-2

Cut or F
Along L