IBM®

NetView™

SC31-6078-1

Customization: Using Assembler

Release 3

File Number
S370/4300/30XX-50

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

Second Edition (August 1989)

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 5665-362) in compatibility mode
VM (NetView program number 5664-204).

Contents

Figures

Tables

About This Book

NetView Customization: Using Assembler describes the ways system programmers can tailor or supplement the NetView™ program, to satisfy unique requirements or operating procedures.

This book discusses the uses and advantages of user-written programs (exit routines, command processors, and subtasks). It provides instructions that guide the programmer through the mechanics of designing, writing, and installing these programs.

This book contains product-sensitive programming interfaces provided by NetView. Installation exits and other product-sensitive interfaces are provided to allow the customer installation to perform tasks such as product tailoring, monitoring, modification, or diagnosis. They are dependent on the detailed design or implementation of the product. Such 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.

Who Should Use This Book

This book is intended for experienced system programmers who are knowledgeable in assembler language. These users are presumed to be familiar with the functions NetView provides automatically.

Secondary users include operators, network planners, designers, and systems analysts. Secondary users may also include IBM marketing representatives and instructors.

What Is New for NetView Release 3

Although the assembler-language programming interface described in this book is not new for NetView Release 3, this book documents the interface in a more useable manner. In addition, the following list of changes and additions have been made in assembler-language support provided by NetView Release 3:

- Templates are provided in the NetView sample library for a user exit routine, a command processor, and an optional subtask. See "Template for a User Exit Routine" on page 48, "Template for a Command Processor" on page 71, and "Template for an Optional Task" on page 95.
- Several sample command processors and user exit routines are provided in the NetView sample library. See Appendix A on page 225.
- In order to avoid possible conflicts with user-written exit routines, null exit routines for DSIEX02A and DSIEX16 are no longer provided.

™ NetView is a trademark of International Business Machines Corporation.

- Information is provided in Chapter 6 on writing REXX user functions, and the DSIRXEBS macro is provided to support this process. See Chapter 6 on page 109 and the DSIRXEBS macro on page 213.
- The DSIRXCOM macro is provided to support access to variables in a calling REXX command list. See "DSIRXCOM – Access REXX Variables" on page 207.
- A PRIORITY option has been added to the DSIMQS macro. See "DSIMQS – Message Queuing Services" on page 185.
- The SUB option has been added to the DSIPRS macro to request the parse service routine to accept quoted substrings. See "DSIPRS – Parsing Services" on page 193.
- A COMPCDE option has been added to the DSIPOP macro. See "DSIPOP – Remove Long Running Command" on page 190.
- The ROLL option has been changed and the PROMOTE option has been added to the DSIPUSH macro. See "DSIPUSH – Establish Long Running Command" on page 202.
- The DSIWLS macro can now be used to write to a sequential log. See"DSIWLS – Write Log Services" on page 214.
- Focal point alert forwarding is now provided. The cross-domain alerts can be accessed in an XITCI exit routine running under the BNJDSERV DST. See "XITCI: CNM Interface Input" on page 44.

Terms Used in This Book

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 list and command processor 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.
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 supported are PL/I and C.
NetView Command List Language	An interpretive language unique to NetView that is used to write command lists.
REXX	Restructured Extended Executor. 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).

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 269.

Evaluation and Education	
Network Program Products General Information	GC30-3350
Bibliography and Master Index for NetView, NCP, and VTAM	GC31-6081
Learning about NetView: Operator Training (PC Diskettes)	SK2T-0292
Planning	
Network Program Products Planning	SC30-3351
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
Operation	
NetView Operation Primer	SC31-6020
NetView Operation	SC31-6019
NetView Command Summary	SX75-0026
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

Chapter 1. Assembler Application Program Interface (API)

Before reading this chapter, you should have read the section on designing user-written functions and the NetView™ customization facilities in the *NetView Customization Guide*.

To use this manual most effectively, you should have in mind a specific command processor or user exit that you want to write in assembler. For example you might want to write a command processor to run under a Data Services Task (DST) to store some data in a VSAM file or a command processor to run on an Operator Station Task (OST) to display information on an operator's screen. The *NetView Customization Guide* contains information to help you decide which command processors and user exits you need to write in order to build your application and what the appropriate language is for each of those pieces of code.

Why Write in Assembler Language?

Although more effort is required to code in assembler language, this language, by its very nature, has greater flexibility than other languages supported by NetView:

- *Regression support:* Your assembler language command processors from earlier releases of NetView, with or without modification, will be supported.
- *Display Flexibility:* Although full screen display support is provided to all languages via the VIEW command, you may wish to take advantage of special features of your terminal hardware, such as wide screens or cursor dependent function. By writing in assembler language you gain direct access to the 3270 data stream commands.
- *Special Data Handling:* Assembler language affords you the ability to do special handling, such as reentrant updates to a global data structure, or complex functions at operator logoff or ABEND-reinstatement.
- *Privileged Functions:* In assembler language, you have unrestricted access to all system functions and macros.
- *Interaction with other commands:* In assembler language, you have the ability to wait on, or examine the status of, asynchronous events other than the standard set that the VIEW command uses. This would include completion of commands under another task or timed events.
- *Performance:* While high-level language routines supported by NetView run faster than interpreted command lists, some kinds of data manipulation can be done even faster at the assembler-language level.

Additionally, you will want to understand the requirements and environments of command processors at the assembler level if you write assembler language subroutines for your high-level language command processors.

™ NetView is a trademark of International Business Machines Corporation.

Overview of the Chapters

The following describes the chapters in this manual.

- Chapter 2. Designing Your Assembler Module - This chapter discusses the information needed to design your assembler module.
- Chapter 3. Writing User Exit Routines - This chapter discusses how to design, code and install user exit routines that take advantage of the NetView program exits at strategic processing points.
- Chapter 4. Writing Command Processors in Assembler Language - This chapter discusses how to design, code and install assembler-language command processors for NetView.
- Chapter 5. Writing User Subtasks - This chapter discusses the two methods for writing user subtasks.
- Chapter 6. Writing REXX User Functions - This chapter explains how to add additional functions or expand or replace REXX functions that already exist in the NetView/REXX environment.
- Chapter 7. Control Block Reference - This chapter describes control block fields related to customization interfaces.
- Chapter 8. Macro Reference - This chapter describes the purpose and coding of the NetView program macros.

Preparing Your Code for Use

Before putting your code into production, develop test scenarios for the major functions. Use the NetView TRACE facility when testing so that you can later analyze any errors or ABENDs your code generates. The format of the TRACE command is found in *NetView Operation*. Running TRACE is particularly helpful for verifying input to user exit routines, to DSIPSS, and to DSIMQS.

When testing new command processors, use RES=N in the CMDMDL definition so that the object code can be replaced without recycling NetView.

If you have included automated operation support in your user-written code, you may test the results of automation using any of the following methods:

- Log on to NetView using an automated operator task user ID and password (your autotask will be treated as an operator task when directly logged on) and observe the output on the screen.
- Examine the network log, the MVS system log, and the hard copy task output for messages associated with the autotask's user ID.

To avoid possible conflict between your code and NetView services currently in use, consider running the code on a separate NetView or test machine, or during a time of day when NetView is not used heavily. To test your code, you must restart NetView, except for pre-defined non-resident command processors. (If there is little risk of conflict, you may simply add your code to the NetView production library currently in use.)

Once the test system is up, perform the test scenarios and verify that the results and output of your code are correct. If they are correct, put the code into production and begin using it normally.

Chapter 2

Chapter 2. Designing Your Assembler Module

This chapter presents information on NetView services available for user-coded command processors, user exits, and user subtasks. Refer to Chapter 8 on page 161 for detailed information on the NetView service macros that are discussed. See Appendix A on page 225 for assembler coding samples incorporating these macros.

Task Structure

The NetView task environment consists of a main task and six types of subtasks.

MNT The main task initializes NetView. It provides an environment for the subtasks and oversees their creation and cleanup. The main task also provides limited exit routines that modify processing under the main task.

OST An operator station task lets an operator enter commands and receive network messages. An OST is created for an operator at logon.

The unattended operator task is a special type of OST that is started by the AUTOTASK command. This task functions as an OST with no associated LU. Independent of VTAM or terminals, this task performs all line-mode commands (no full-screen commands) and allows multiple command procedures to run concurrently in separate subtasks. The unattended operator task is a basis for automation.

The MVS console operator task is another special type of OST that is started by the AUTOTASK command with the CONSOLE operand. Independent of VTAM, this task performs all line-mode commands (no full-screen commands) and allows multiple MVS console operators to access NetView, each under their own MVS console operator task.

NNT A NetView-NetView task allows an operator to access another NetView (usually in another domain), send commands to that NetView, and receive messages from that NetView. The START DOMAIN command causes an NNT to be created in the other domain.

HCT A hard-copy task logs the activity of one or more OSTs using 328x printers. The START HCL command creates an HCT.

PPT The primary programmed operator interface (POI) task receives unsolicited messages from VTAM and receives message traffic exchanged between the system console and VTAM. Additionally, the PPT provides timer services and runs commands and command lists. There is one PPT per NetView. The PPT is created when NetView starts.

DST Data services tasks provide command, message, and exit functions. In addition, data services tasks can manage VSAM or CNM facilities. You can create new subtasks to perform any or all of these functions. The DST is created when NetView starts or when START TASK is issued.

OPT Optional tasks provide increased flexibility beyond the subtasks NetView provides. For example, an OPT could be used to provide unique command interfaces or more precise control of work dispatching than standard command processors running under a DST-based task could provide. You can define the OPT to start when NetView starts or when START TASK is issued.

General Coding Guidelines

Processing Environment

User written code of all types is given control in problem program state and with a user memory protection key[1]. NetView and all user written code given control under NetView is authorized. This gives the user the option of invoking privileged system services or entering supervisor state or changing the key. However, control should be returned to NetView in the same program state and key as when user code was entered.

Except as noted in Chapter 7, all NetView control blocks are accessible in the user key. Unless specifically noted in Chapter 8, all NetView services must be invoked in problem program state and the user key.

The setting of the TVBINXIT bit in the task vector block (TVB) when NetView calls user-written code determines the environment under which your code will be processed. When a NetView task is initially dispatched, the TVBINXIT bit is off. Thus, if the task calls user-written code at this time, the code processes under the mainline environment.

Whenever the operating system interrupts mainline processing, NetView sets the TVBINXIT bit on. Thus, any user-written code called at this time is processed under the TVBINXIT=on environment.

Not all tasks can run all types of user-written code. In fact, each task has unique restrictions that limit the code it can process. Table 1 indicates the types of code that can run under each task in the mainline environment. The figure also indicates which types of code can run when the TVBINXIT bit in the task vector block (TVB) is set on. (Since you define which commands and functions will run under an OPT subtask, that subtask is not included in Table 1.)

Table 1. Processing Environment of User-Written Programming

	Main Task	**OST**	**NNT**	**HCT**	**PPT**	**DST**	**TVBINXIT Set On**
User exit routines	✓	✓	✓	✓	✓	✓	✓
Regular command processors		✓	✓		✓		
Immediate command processors		✓	✓				✓
Data services command processors						✓	

Coding Requirements

This section discusses the general restrictions that you must observe when writing your code. The control blocks and macros mentioned in this section are described in more detail in Chapter 7 and in Chapter 8.

[1] Unless your system programmers set up special provisions, this will be key 8 in MVS and key 14 in VM.

Restrictions when TVBINXIT is On: See Table 2 on page 32 to determine which user exits must interrogate the TVBINXIT bit. No user exit should change the value of TVBINXIT. When your code receives control with TVBINXIT set on, the following restrictions apply:

- Messages may be sent only to the immediate message area of the operator's screen. To send these messages, you must specify macro DSIPSS with the TYPE=IMMED option. You may also use DSIMQS to queue messages to be processed after the asynchronous exit completes.
- Do not code your program to cause a system wait state. A system wait may produce an interlock. For example, if an IRB exit were to wait for a resource being used by the mainline routine, the resource would not be released since the IRB exit is interrupting the mainline routine.

  To avoid causing a system wait state, keep in mind the following:

  - Do not issue macro DSIWAT.
  - Do not issue macro DSIDKS or any other disk services macros. Disk I/O can cause a wait state.
- Only immediate commands may be called directly. DSIMQS can be used to invoke regular commands. For more information, see "Invoking Commands" on page 21.

Variable Names: Do not use the following as a prefix for any variable or message you name in your code:

- Any NetView component prefix, AAU, BNI, BNJ, BNK, CNM, DSI, or DWO.
- Any NetView control block suffix, such as SWB or PDB.

Macro Usage: When a NetView macro and an operating system macro perform equivalent services, use the NetView macro. This ensures that the source code for your program will be transportable across NetView operating systems. However, if you must use an operating system macro, be careful that its function does not conflict with a function NetView may be performing. For example, if the operating system macro STIMER were issued under the PPT, NetView timer services would be disrupted.

Register Usage: Follow these guidelines when using registers for user exit routines, command processors, and subtasks:

- Save registers at entry to the user code and restore them before returning control to NetView. Set register 15 with your return code.
- Avoid using registers 0, 1, 14, and 15; they are reserved for NetView macro expansion.
- Register 13 should contain the address of a standard 72-byte save area.
- Return control to the location in NetView specified by register 14.
- Do not rely on the contents of registers 0 and 2 through 12 for constant values on entry to user code. Their contents vary.

Reentrancy: Make sure your code is reentrant, to allow the one copy of the program to be used concurrently by more than one task. For example, a command processor may be invoked from two or more tasks simultaneously and thus execute simultaneously under multiple tasks.

Reentrancy is not required for certain user exits or for optional tasks. SeeTable 2 on page 32 for more information.

MVS/XA Considerations: NetView supports any valid combination of addressing mode (AMODE) and residency mode (RMODE). Depending on the mode your code is currently running in, NetView provides the appropriate services. In addition, it provides macro parameters where necessary to enable you to specify 24-bit or 31-bit addressing for your special requirements. Also, NetView provides appropriate residency for the control blocks your code accesses, such as USE, CWB, and BUFHDR. For example, if your command processor is loaded with AMODE=24, then the CWB passed to it will reside below 16 Mb.

To conserve storage below 16 Mb, 31-bit addressing is recommended except when restricted by your use of MVS/XA interfaces to the contrary.

Control Blocks

To perform the essential functions of presenting information to end users and invoking commands, your code must include the necessary control block mappings and establish addressability to other control blocks. For details on the purpose and contents of the control blocks mentioned below, refer to Chapter 7 on page 119.

Control Blocks Overview

- MVT - Main Vector Table (DSECT DSIMVT)

  One MVT for NetView, contains NetView global information.

- TVB - Task Vector Block (DSECT DSITVB)

  One TVB per task, contains task definition and status information.

- TIB - Task Information Block (DSECT DSITIB)

  One TIB per active task, contains additional information pertaining to a task.

- SWB - Service Work Block (DSECT DSISWB)

  Work area/parameter list for NetView macro services. An SWB is provided at entry to a command processor or user exit.

- CWB - Command Work Block (DSECT DSICWB)

  One per command, contains savearea, work area, and pointers to other NetView control blocks.

- PDB - Parse Descriptor Block (DSECT DSIPDB)

  In a command processor environment, the PDB contains parse information for the command. In a user exit environment, the PDB contains parse information for the user message buffer.

- BUFHDR - NetView Buffer Header (DSECT BUFHDR included with DSECT DSITIB)

  Maps the control information which precedes commands and messages in NetView buffers.

- IFR - Internal Function Request (DSECT DSIIFR)

  A multi-purpose NetView buffer:

  - Command IFR (contains a command)

  - Automation IFR (contains automation information and message(s))

- USE - User Exit Parameter List (DSECT DSIUSE)

  Contains user exit information.

- DSRB - Data Services Request Block (DSECT DSIDSRB)

  CWB extension used by Data Service Command Processors in the DST (Data Service Task) environment only.

Including Control Blocks: At assembly, include DSECTs for the NetView control blocks that have fields your code uses. You must include the following control blocks in your code:

- For user exit routines, include the user exit parameter list (USE). Since the USE contains the addresses of the TVB, SWB, PDB, and BUFHDR (defined with the TIB), you need to include these control blocks, as well.

  Note: Each TVB will address its associated TIB and the MVT.

- For command processors, include the command work block (CWB). Since the CWB contains the addresses of its SWB, PDB, and BUFHDR (defined with the TIB), you also need to include these control blocks. In addition, data services command processors (DSCPs) require the data services request block (DSRB).

- For user subtasks, include the task vector block (TVB). Then include any other control blocks necessary for your user-defined task.

Depending on the particular service facilities your code uses, you will need to include various other control blocks.

Figure 1 on page 12 illustrates the interconnections between the control blocks.

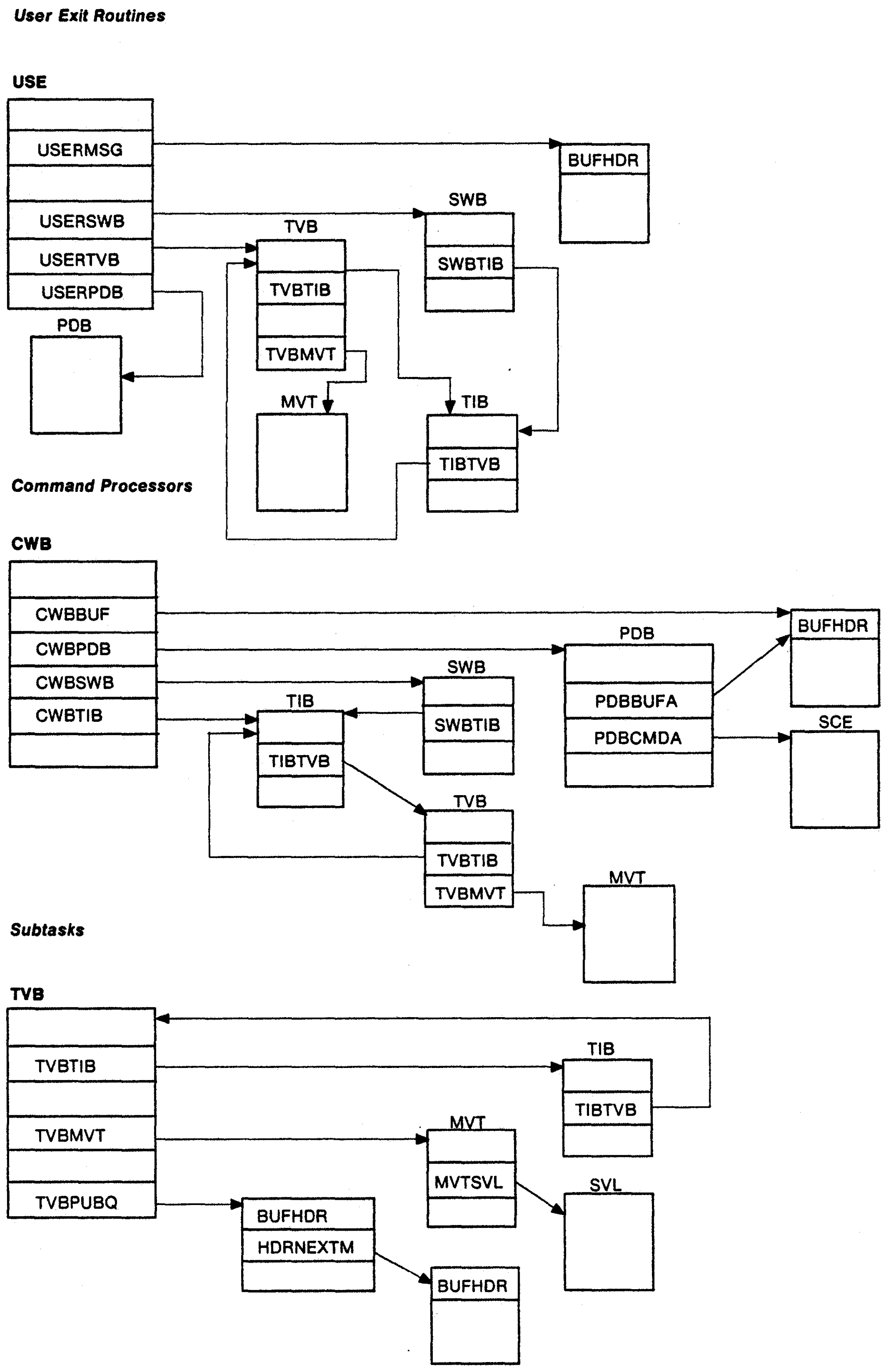


Figure 1. Overview of the Control Blocks for User-Written Programming. For each type of user-written code, the TVB provides access to the MVT and, thus, to the SVL, as shown in the subtasks diagram. The field displacements suggested in the figure are not representative of the actual field displacements.

Use macro DSICBS to include DSECTs for the appropriate control blocks. For example, you might code the DSICBS macro in a command processor as follows:

```
DSICBS DSICWB,DSISWB,DSIMVT,DSIPDB,DSITVB,PRINT=NO
```

This instruction includes the CWB, SWB, MVT, PDB, and TVB. This instruction also specifies, with PRINT=NO, that control block expansions are not to be printed.

Establishing Addressability: You must establish addressability to a control block before referencing its fields. The following example shows how you could establish addressability to the MVT for a command processor:

```
USING DSICWB,1
L     register,CWBTIB
USING DSITIB,register
L     register,TIBTVB
USING DSITVB,register
L     register,TVBMVT
USING DSIMVT,register
```

Work Block Services

The service work block (SWB) is needed when your code invokes NetView service facilities. The command work block (CWB) is used as command processor input. Macro DSILCS obtains and releases SWBs or CWBs, as shown in the following example:

```
DSILCS CBADDR=MYSWBPTR,SWB=GET
LTR    REG15,REG15
BNZ    ERRORSWB
L      REG2,TVBTIB
L      REG3,MYSWBPTR
ST     REG2,SWBTIB-DSISWB(REG3)
   :
DSILCS CBADDR=MYSWBPTR,SWB=FREE
```

In this example, an SWB is to be obtained and its address is to be placed in MYSWBPTR. The second and third statements test for a good return code in register 15. If the macro was successful, you must initialize the SWBTIB field to your TIB address before the SWB may be used. After use, the SWB should be released using DSILCS.

Basic Module Services

Standard NetView Buffer Structure

All message and command buffers must include an initialized buffer header (BUFHDR) structure preceding the message data or command text. BUFHDR is a separate DSECT contained in the DSITIB (Task Information Block) control block (use DSICBS to include DSITIB when your module references BUFHDR fields). The following section presents an overview of the BUFHDR fields. See Chapter 7 on page 119 for additional details.

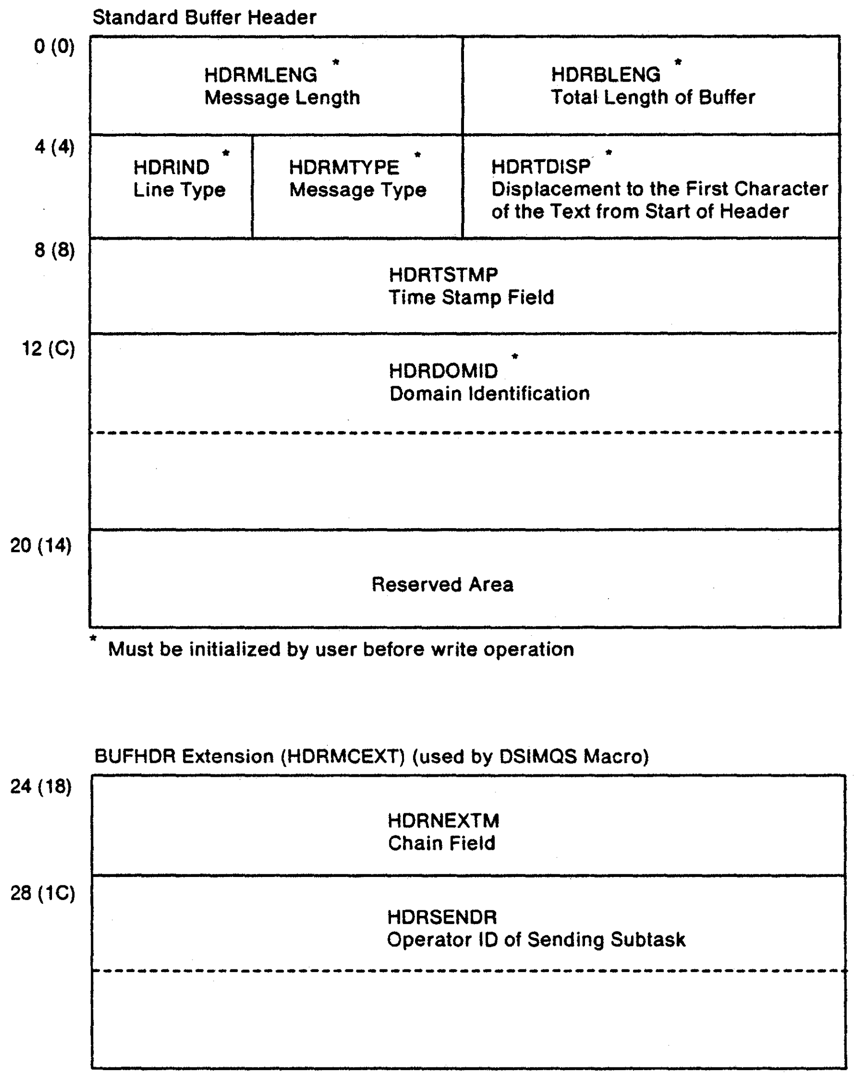


Figure 2. Buffer Header (BUFHDR)

Field	Description
HDRBLENG	Contains the actual length of the entire buffer: header, plus text, plus unused space. If the buffer is to be released with DSIFRE, this length is used. The length may be up to 32,767 bytes.
HDRMLENG	Indicates the length in bytes of the text data in the buffer.
HDRIND	Should be set to zero except when using title-line output. See "Title-Line Output" later in the chapter for details.
HDRMTYPE	Contains a character that indicates the current usage of the buffer. It may also indicate the origin of the command. If the buffer is written out using the DSIPSS macro, this field is displayed and logged.
HDRTDISP	The offset from the start of the buffer header to the first byte of text.
HDRTSTMP	Contains the time that the command was received, in the packed decimal from X'*hhmmss*0C' where *hh* is the hours of the day from 00 to 23, *mm* is the minutes of the hours from 00 to 59, *ss* is the seconds of the minute from 00 to 59, and 0C is a packed decimal sign. See the DSIDATIM macro instruction.
HDRDOMID	Shows the identifier of the domain where the message originated. This field is displayed and logged.
HDRMCEXT (BUFHDR extension)	An extension to the BUFHDR that is used when a buffer is transferred from one subtask to another. It is built by the DSIMQS macro when creating a buffer for the destination task. Other buffers do not need these fields.
HDRNEXTM	An internal NetView field that is used to chain buffers together.
HDRSENDR	Contains the originator's operator ID, which is the contents of the sender's TVBOPID field.
TEXT	Can start anywhere after the reserved area in a standard buffer or after HDRSENDR in a buffer with a message command extension. Use HDRTDISP to locate the start of text.

Note: Relationship of HDRBLENG, HDRMLENG, and HDRTDISP: Since HDRTDISP points to the start of the buffer data and HDRMLENG is the length of the buffer data, then at no time should HDRTDISP + HDRMLENG be greater than HDRBLENG. This implies that the value of HDRMLENG can range in value from 0 to (HDRBLENG − HDRTDISP).

Dynamic Storage Services

Storage services enable you to get and free storage for your code. Use macro DSIGET to get storage and macro DSIFRE to free storage, as shown in the following example:

```
DSIGET LV=4096,A=STORPTR,TASKA=(MYTVBREG),Q=NO
LTR    REG15,REG15
BNZ    ERRORGET
   :
DSIFRE LV=4096,A=STORPTR,TASKA=(MYTVBREG),Q=NO
```

The above example requests that 4,096 bytes of storage be obtained. The address of the storage is to be placed in the fullword named STORPTR. The second and third statements test for a good return code in register 15 before you use the storage. When the storage is obtained, it has been cleared to zeros.

After use, you must free the storage using the same value for the Q option as you used previously to get the storage.

Note: The use of TASKA is recommended, since it can help you avoid addressing the wrong TVB control block. It must contain the address of the TVB for the subtask where the code is executing. Use the TASKA parameter for both non-queued (Q=NO) and queued (Q=YES) storage.

Queued Storage: You may also use the DSIGET Q=YES option to enable NetView to release queued storage at logoff or task termination, which facilitates error recovery. The following example illustrates this usage:

```
DSIGET LV=4096,A=STORPTR,TASKA=(MYTVBREG),Q=YES,BNDRY=PAGE
LTR    REG15,REG15
BNZ    ERRORGET
   :
DSIFRE A=STORPTR,TASKA=(MYTVBREG),Q=YES
```

In this example, Q=YES indicates that NetView is to keep track of the 4,096 bytes of storage in an internal queue. To support this internal queue, Q=YES generally gets eight more bytes of storage than requested. However, if BNDRY=PAGE is specified, eight extra bytes are not gotten and thus page alignment is not affected.

Storage obtained with one call must be released with one call.

NetView ignores the storage length indicated by DSIFRE when Q=YES is specified. LV is optional when Q=YES and is ignored.

Message Processing

Creating Messages

The DSIMDS (Message Definition Service) macro provides the ability to create a load module of user defined message skeletons. Each defined message skeleton may contain up to nine variable length text inserts.

The DSIMBS (Message Building Service) macro will take message insert text and combine it with the specified message skeleton and return a completed message buffer which can be used for displaying the message.

See Chapter 8 on page 161 for additional details on using the DSIMDS and DSIMBS macros.

See samples CNMS4271, CNMS4278, and CNMS4281 for an example of creating user defined messages with DSIMDS and DSIMBS.

Displaying Information to NetView Operators

The primary channels for presenting information to the operator are the following:

- The terminal screen using macro DSIPSS
- The network log, MVS system log, sequential log, and hard-copy log using DSIWLS
- The system console using DSIWCS.

Figure 3 shows how these macros communicate with the operator. The logging and system console services are simply invocations of the DSIWLS and DSIWCS macros. For detailed information on these services, see Chapter 8 on page 161. Message presentation via DSIPSS and DSIMQS is more complex and is described on the following pages.

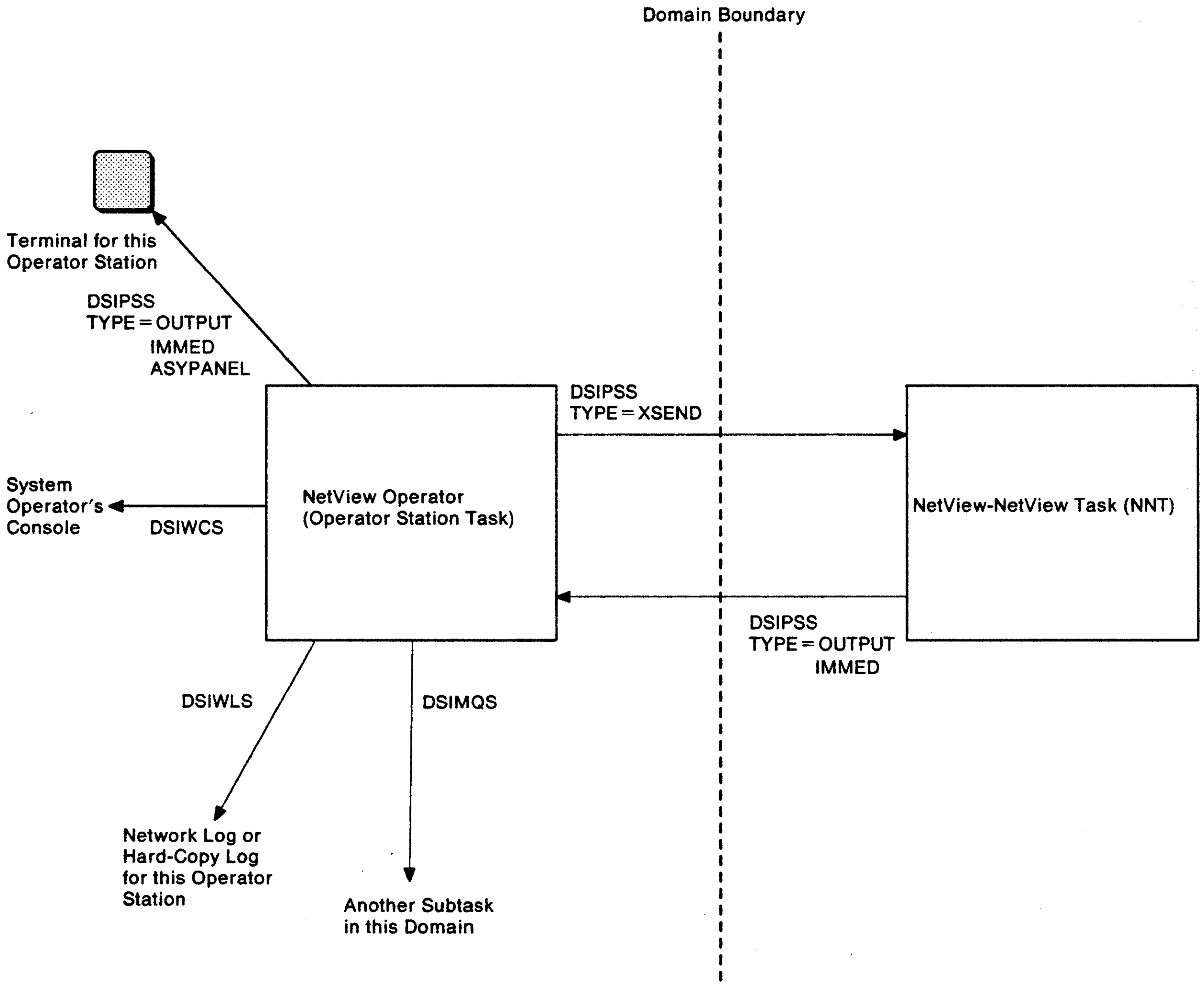

Domain Boundary
Terminal for this Operator Station
DSIPSS TYPE=OUTPUT IMMED ASYPANEL
DSIPSS TYPE=XSEND
System Operator's Console
DSIWCS
NetView Operator (Operator Station Task)
NetView-NetView Task (NNT)
DSIPSS TYPE=OUTPUT IMMED
DSIWLS
DSIMQS
Network Log or Hard-Copy Log for this Operator Station
Another Subtask in this Domain

Figure 3. Using Macros to Communicate from an OST

Macro DSIPSS can present information in any of the following three screen modes; DSIMQS is limited to standard mode and title-line mode.

Standard Mode From an OST or NNT, messages are sent to the screen with a 12-character prefix followed by data. The prefix includes a 1-character code for the entry type (from HDRMTYPE) and a domain name field (from HDRDOMID) indicating the domain that generated the message. If the message exceeds the screen width, it is split between two words. The message is continued on the next screen line and indented 12 characters.

A variation on standard mode output is the immediate message. It appears at the bottom of the screen as a single 71-character message with neither prefix nor continuation lines.

Thus, to present a message in standard mode, you may use any of these methods:

- Issue DSIPSS TYPE=OUTPUT
- Issue DSIPSS TYPE=IMMED
- Issue DSIMQS to queue a HDRTYPEU message buffer to the OST.
- Issue DSIPSS TYPE=FLASH

See sample CNMS4274 for an example of standard mode output.

Title-Line Mode

Title-line presentation services send sequences of messages to the operator, without allowing other messages to be interspersed. The messages appear on the screen in a tabular format, with one or more title lines. Title-line messages have no prefix and may use the full width of the screen. Messages longer than screen width are truncated. For more information on how to specify this type of output, see "Title-Line Output" on page 19.

Title-line mode messages and system multiline write-to-operator (MLWTO) messages are treated as a single message by DSIEX02A, DSIEX16, &WAIT in NetView command list language, TRAP and WAIT in REXX and high-level language command procedures, the message automation table, and the ASSIGN command. When creating your title-line mode messages, make sure the first line does not conflict with any IBM-supplied message number. Some message number format similar to the IBM message number format is recommended to allow these facilities to be used with your messages.

See sample CNMS4273 for an example of title line output.

Full-Screen Mode

Application-built 3270 data streams containing commands, orders, and data are sent to the screen. In this way, a full screen of information can be presented. Full-screen command processors, which run under an OST, are the only type of user-written code (except for user exit DSIEX12) that can utilize full-screen mode. For this reason, this topic is addressed more fully under "Writing a Full-Screen Command Processor" on page 61.

See sample CNMS4279 for an example of full screen output.

Title-Line Output

Title-line output is best suited to message groups that can be presented on a single screen, since you cannot scroll backward. Figure 4 shows an example of title-line output.

```
NETVIEW                                          mm/dd/yy hh:mm:ss
=NETV1
NCP    LINE    PU/CLUSTER    LU/TERMINAL      TYPE    LOCATION
-----  ----    -----------   ------------     -----   ----------

NCPA                                          3705    MACH. ROOM
NCPA   A01                                    SDLC    SATELLITE
NCPA   A01     A01A                           3274    ANCHORAGE
NCPA   A01     A01A          A01A01           3278    ANCHORAGE
NCPA   A01     A01A          A01A02           3278    ANCHORAGE
NCPA   A01     A01A          A01A03           3278    ANCHORAGE
NCPA   A01     A01B                           3274    NOME
NCPA   A01     A01B          A01B01           3278    NOME
NCPA   A01     A01B          A01B02           3278    NOME
NCPA   A01     A01B          A01B03           3278    NOME
 .      .       .              .               .       .
 .      .       .              .               .       .
 .      .       .              .               .       .
 .      .       .              .               .       .
 .      .       .              .               .       .
```

Figure 4. Example of Title-Line Output

The first line NetView generates is a separator that consists of just the message type and the domain ID. (In Figure 4, "=" is the message type and "NETV1" is the domain ID.) The separator is followed by the title, which can be one to six lines. The title is followed by the data lines. The screen wraps around until all the data is displayed. (If the data lines continue to the next screen, the title is redisplayed at the top of the new screen.)

To use title-line output, format and send one message buffer for each line of information as follows:

1. Set the HDRMTYPE field in the BUFHDR to HDRTYPEL (=).
2. As you build each message buffer, do the following:
   - For a line that is part of the title, set HDRIND to HDRLNLBL. Your title can contain one to six lines.
   - For a data line, set HDRIND to HDRLNDAT.
   - For the last line of data, set HDRIND to HDRLNEND.
3. When the message buffer is ready for presentation, do one of the following:
   - From any NetView subtask, use DSIMQS to queue the message buffer to the desired OST. This is the recommended method and is appropriate to use from the destination OST; however, the output will not begin to be displayed until you return control to the OST.
   - From an OST or NNT, not in an exit, issue DSIPSS TYPE=OUTPUT to send the message buffer.

In either case, the output will not begin to be displayed until after the data-end line has been sent.

NetView groups all title-line buffers at the OST until a buffer marked as data end (HDRLNEND) is received. Upon receipt of the end message, the title lines are sent to the screen. These lines appear directly under the previous message. Each data line appears one line at a time. If the message sequence fills one screen and begins another, the title lines are repositioned at the top of the screen, followed by the next data lines. This process continues until all the messages appear on the screen.

Each buffer sent to the screen contains at least one character. To print a blank line, place a blank character (X'40') in the buffer. If a line of title-line output is longer than the width of the screen, the line will be truncated to screen width.

Displaying Messages to the System Console

The DSIWCS (Write to Console Service) macro will display a message buffer at the system console. The message buffer must have an initialized NetView buffer header (BUFHDR) and the text will be truncated to 120 characters when displayed on the console.

Logging Messages

The DSIWLS (Write to Log Service) macro can be used to record information on the network log, the MVS system log, a hard copy log, an external log, or a sequential log.

Network Log, System Log, and Hard Copy Task: DSIWLS will log data to the network log, system log, and the hard-copy task. The DEFAULTS and OVERRIDE commands will determine the general logging environment (the current logging actions will be applied to the message buffer passed to the DSIWLS macro). If the message is passed in an AIFR buffer, the AIFR settings will be checked to determine which of the logs the message will actually be logged in.

See sample CNMS4272 for an example of writing to the logs.

External Logging: DSIWLS also provides the ability to log data to the NetView-defined external logging task (task ID is DSIELTSK). The external logging task can be defined at installation time to record to the SMF log (restricted to MVS) or to a user defined data set. If a user defined dataset is to be used, the XITXL exit must be coded to actually do the logging of the data. The external logging task is a NetView-implemented Data Services Task (DST). See Chapter 8 on page 161 for information on DST's and refer to the *NetView Installation and Administration Guide* for details on installing the external logging task. The XITXL exit is described in Chapter 3 on page 31.

Sequential Logging: DSIWLS also provides the ability to log data to a sequential logging task. The sequential logging task will record the data using the Basic Sequential Access Method (BSAM). Multiple sequential logging tasks can be defined at installation time. Refer to the *NetView Installation and Administration Guide* for details on installing sequential logs.

See sample CNMS4275 for an example of writing to a sequential log.

Invoking Commands

Commands can either be called directly or they can be scheduled. Calling a command directly requires the following:

1. The command environment must be initialized (required control blocks must be acquired, etc.).
2. The DSICES (Command Entry Service) macro must be invoked to look up the address of the command processor in the NetView command table and then a branch is performed to the command processor.

To schedule a command, the DSIMQS (Message Queuing Service) must be used. Scheduling a command under a task results in a command buffer being placed on one of the task's message queues (there may or may not be other command buffers ahead of the scheduled command's). When the scheduled command's buffer is processed off the message queue, the command processor will be invoked.

Calling a Command Directly

The following steps must be followed to call a command processor directly:

All the following are required to invoke a command processor or command list:

- A CWB
- An SWB
- A command buffer
- A PDB
- A save area
- Registers 1, 13, 14, and 15.

Obtaining a Command Work Block (CWB): A command processor requires a command work block (CWB) for use as a parameter list, a save area, and a work area. A CWB may be preallocated (and reused) or may be obtained by issuing the DSILCS macro. Before calling the command processor, the TIB address must be stored in the CWBTIB field.

Obtaining a Service Work Block (SWB): The routine that invokes a command processor must provide an SWB. The SWB may be preallocated, obtained with the DSILCS macro, or may be one the invoker was passed. This control block must also have its SWBTIB field pointing to the TIB. The SWB address must be stored in CWBSWB.

Building a Command Buffer: Each command is invoked with a command buffer containing a verb and optional operands. The command buffer is prefixed with the buffer header (BUFHDR). Each BUFHDR field must be initialized except the message command extension HDRMCEXT. The address of this command buffer is stored in CWBBUF.

Obtaining a Parse Descriptor Block (PDB) and Parsing the Command: The routine must obtain storage for a PDB to parse the command for the command processor to be called. The size of the storage for the PDB may be obtained by issuing the DSIPRS macro with the PDBSIZE option. The usual size is 160 bytes. After the storage is obtained (from preallocated storage or with DSIGET), the address is stored in the CWBPDB field. The control block header (CBH) is built and CBHID is set to the value defined by symbol CBHPDB. CBHTYPE is zeroed, and the PDB length is stored in the CBHLENG prior to invoking the DSIPRS macro. Issuing DSIPRS fills in the PDB including the PDBBUFA pointer to the command buffer, the parse elements, and the number of parse elements.

Looking Up the Command Processor Address: After the command is parsed, the command must be found in the NetView system command table (SCT). The command may be looked up in one of three ways:

- With the parsed command in the PDB
- Without prior parsing
- By command processor module name (the module name is known but the verb name may change, as in a synonym).

The DSICES macro returns the address of the command's system command table entry. This address is returned in an area passed on the DSICES macro as the SCTADDR parameter. This address points to an SCT entry (mapped by the SCE), and the address must be stored in the PDBCMDA field.

When the DSICES macro returns to the caller, the return code indicates whether the command is immediate, regular, or both. If TVBINXIT is on, then the PDBIMMED bit must be set on if the DSICES return code indicates that the command to be called was defined as TYPE=I (immediate) or TYPE=B (both).

You may call regular commands only when TVBINXIT is off and only from task types OST, NNT, and PPT. You may call immediate commands only when TVBINXIT is on and only from task types OST and NNT. Unless otherwise noted in this book, do not call NetView-written TYPE D commands.

Calling the Command Processor: Register 1 must point to the CWB (which now in turn points to the PDB, SWB, TIB, and the command buffer). Register 13 must point to a save area (where it is probably already set, because a save area is required for the service macros). Register 15 must contain the command processor's entry point address (found in SCE) and register 14 must have the return point address. The command processor entry point address is stored in the SCECADDR field of the SCE entry pointed to by the PDBCMDA field.

For example, to pass a command to VTAM while running under OST, NNT, or PPT, prepare the input described above. Call the NetView command processor identified by DSICES for your VTAM command.

See sample CNMS4280 for an example of calling a command directly.

Scheduling a Command Using DSIMQS

Simulating Terminal Input: The simplest way for user-written code to invoke regular commands to run under an OST or NNT is to simulate commands entered from a terminal. Include the command in the text of a standard buffer with an initialized buffer header, as described under "BUFHDR — Buffer Header" on page 121. In BUFHDR, set HDRMTYPE to HDRTYPET. You may use HDRTYPEB as well; however, HDRTYPEB commands will be neither logged nor echoed.

Then use macro DSIMQS to queue the buffer to the desired OST or NNT, where the command will be processed as though it had been entered from a terminal.

Usually, commands from an operator are scheduled with high priority. This allows the command to preempt any existing queued messages or other work that has been scheduled at lower priority. Commands scheduled with high priority will also preempt command procedures that are already executing. If you do not desire to preempt work that may already be queued, including command procedures that are already executing, then you should schedule your command at low priority. See "DSIMQS – Message Queuing Services" on page 185 for more information on priority.

Building an IFRCODCR: To pass commands to a subtask in the same domain, you can build an IFRCODCR and queue it to the desired subtask. An IFRCODCR (see "IFR — Internal Function Request" on page 133) is an internal function request (IFR) that requests that a command be invoked. IFRCODCR is intended for requesting or conveying data. The command driven by this type of buffer should not present data to the operator, neither by full screen nor line mode, and it should not create or remove a long running command. These actions could be disruptive because the operator could be engaged in unrelated activity.

The IFR requires an initialized buffer header (BUFHDR), a message command extension (HDRMCEXT) if DSIMQS BFRFLG=YES is specified, and an IFRCODE set to IFRCODCR. The command and its parameters follow the IFRCODE.

In BUFHDR, HDRMLENG must reflect the length of the command and its parameters, as well as the length of the IFRCODE field. Set HDRTDISP to the offset of the IFRCODE field. HDRMTYPE must be set to HDRTYPEI.

Use macro DSIMQS to send the buffer to any subtask that can process a command. When the buffer is received, NetView will have increased HDRTDISP by 2 to address the command and its parameters. NetView will have decreased HDRMLENG by 2, because the IFRCODE is not included in the command text.

For more details on the fields and settings of BUFHDR and IFR, see "BUFHDR — Buffer Header" on page 121 and "IFR — Internal Function Request" on page 133.

See sample CNMS4283 for an example of scheduling a command.

Additional Module Services

Loading and Deleting Modules in Virtual Storage

Modules which are used infrequently can be dynamically loaded and deleted in virtual storage using the DSILOD and DSIDEL macros. Use DSILOD to load the module and DSIDEL to delete the module.

See sample CNMS4271 for an example of using DSILOD and DSIDEL.

Event Control Block (ECB) Services

Posting and waiting on Event Control Blocks should be done using the DSIPOS (post) and DSIWAT (wait) macros. DSIWAT allows waiting on a single ECB or on a list of ECBS.

See the optional task template (CNMS4277) for an example of waiting on an ECB list.

Disk Services

The disk services macro retrieves data from partitioned data sets (for MVS) or from files (for VM). The macro DSIDKS connects to a data set or filetype, locates a specific member or file, and reads the records there, as illustrated in the following example:

```
DSIDKS  SWB=(REG2),DSBWORD=DISKADDR,TYPE=CONN,NAME=DSIPRF
   :
DSIDKS  SWB=(REG2),DSBWORD=DISKADDR,TYPE=FIND,NAME=MEMNAME
   :
DSIDKS  SWB=(REG2),DSBWORD=DISKADDR,TYPE=READ
   :
DSIDKS  SWB=(REG2),DSBWORD=DISKADDR,TYPE=READ
   :
DSIDKS  SWB=(REG2),DSBWORD=DISKADDR,TYPE=DISC,NAME=DSIPRF
```

In the above example, DSIDKS TYPE=CONN initializes the disk service control blocks and input buffer, returning the address of the DSB in DISKADDR. The DDNAME is DSIPRF (for VM, this parameter specifies the filetype, as in NAME=NCCFLST). Next, using the returned DSBWORD, DSIDKS TYPE=FIND finds the member name MEMNAME and reads the first record. The next two DSIDKS TYPE=READ instructions read the next two sequential records. Finally, DSIDKS TYPE=DISC frees the control blocks and the input buffer.

See sample CNMS4276 for an example of using DSIDKS to read a NetView initialization file.

Parsing

NetView command and message buffers (containing the standard NetView BUFHDR structure) can be parsed using the DSIPRS macro. The DSIPRS macro requires a Parse Descriptor Block (PDB). The size of the PDB can be determined by first issuing the DSIPRS macro with the PDBSIZE option specified. This will return the required size of the block. After you obtain the storage (from preallocated storage or with DSIGET), build the control block header (CBH) and set CBHID to the value defined by symbol CBHPDB. Set CHBTYPE to zero and store the PDB length in the CBHLENG field. Then DSIPRS can be invoked with the supplied PDB to actually parse the buffer. DSIPRS will fill in the PDB with the delimiter and parse information.

See sample CNMS4280 for an example of using DSIPRS to calculate the size and initialize a parse descriptor block.

Scope Checking and Parameter Substitution

The DSIKVS (Keyword and Value Service) can be used to determine whether or not a particular command keyword and value are defined for this task's Operator Class (the OPCLASS statement in the Operator Logon Profile determines the defined operator classes).

The DSIPAS (Parameter Alias Service) can be used to derive the original keyword/value for a command which is entered with parameter aliases. Parameter aliases are defined with the PARMSYN statement. See *NetView Administration Reference*.

See sample CNMS4276 for an example of command scope checking.

Named Storage

A storage environment can be easily created and retrieved across multiple command processor calls using named storage.

Macro DSIPUSH can be used to create a named storage pointer, as shown in the following example:

```
         L     R3,LOCLSTOR              BUFFER AREA FOR PUSH LIST
         USING SWBLRCPL,R3
         XC    0(SWBLRCPU,R3),0(R3)
         MVC   SWBLRCLN(4),=A(SWBLRCPU)  LENGTH OF PUSH LIST
         MVC   SWBLRCNM(16),MYNAME      UNIQUE NAME OF LRC
         MVC   SWBLRCST(4),DYNASTOR     QUEUED STORAGE OBTAINED FOR LRC
         MVC   SWBLRCRE(8),ZEROS        NO RESUME FOR NAMED STORAGE
         MVC   SWBLRCAB(8),MYABEND      ADD ABEND MODULE NAME
         MVC   SWBLRCLG(8),=L'DSILRCR8'   (FOR EXAMPLE)
*                      SWBLRCFG (FLAGBITS)  IGNORED FOR NAMED STORAGE
         L     R4,CWBSWB                   AVAILABLE SWB
           SPACE 3
         DSIPUSH SWB=(R4),LIST=(R3),ROLL=NO
```

Then macro DSIFIND can be used to retrieve the named storage pointer, as shown in the following example:

```
         L     R3,LOCLSTOR              BUFFER AREA FOR PUSH LIST
         USING SWBLRCPL,R3
         XC    0(SWBLRCFI,R3),0(R3)
         MVC   SWBLRCLN(4),=A(SWBLRCFI)  LENGTH OF FIND LIST
         MVC   SWBLRCNM(16),MYNAME      UNIQUE NAME OF LRC
         L     R4,CWBSWB                   AVAILABLE SWB
           SPACE 3
         DSIFIND SWB=(R4),LIST=(R3)
         LR    R3,R1                    ADDRESS OF MY NAMED STORAGE
         USING MYDSECT,R3
```

For more details on the coding of these macros, see "DSIPUSH – Establish Long Running Command" on page 202 and "DSIFIND – Find Long Running Command Storage" on page 169. For a discussion of long running command processors, see "Writing a Long Running Command Processor" on page 64.

Scheduling Commands in a Cross-Domain Environment

The DSIPSS (Presentation Services) macro also provides the ability to schedule commands in a cross-domain (OST/NNT) environment:

You can forward a command from one domain to another by doing either of the following (providing the route has been previously established via the START DOMAIN command):

- Building a buffer with the desired command and issuing macro DSIPSS with TYPE=XSEND to transmit the command to the cross-domain task (NNT) in another NetView. (The command runs in the other NetView under an NNT.)

- Calling the ROUTE command. (See "Simulating Terminal Input" on page 22.) The ROUTE command routes a command to a specified NetView domain. For more information, see *NetView Operation*.

The commands you invoke can return data to the originating domain by issuing DSIPSS TYPE=OUTPUT for a buffer with HDRMTYPE=HDRTYPEU or HDRMTYPE=HDRTYPEL.

The maximum length of text that can be sent as a cross-domain command is 240 bytes, which corresponds to three 80-character input lines. Use multiple commands to chain data for applications that require larger data transfer.

Returning a Command to an Originating Domain

For a command running under an NNT to invoke a command in the originating domain, it must issue DSIPSS TYPE=OUTPUT for a buffer with HDRMTYPE=HDRTYPEX. The buffer must contain the desired command verb and parameters. The verb must be delimited from the data or parameters by a blank and must be a defined command in the receiving domain.

For example, if data formatting is required, you can build a buffer with HDRMTYPE=HDRTYPEX and a command in the buffer text. In this case, the command verb identifies a user-defined presentation services command processor and the parameters are the data to be presented. When the receiver of the OST's cross-domain message gets the command buffer, the OST calls the command processor with the data.

Applications are limited to sending 256 bytes of data. Use multiple commands to chain data for applications that require larger data transfer.

Resource Span Checking

You may want a command processor to use NetView span checking facilities to limit operators to particular sets of resources. The following section describes the macros and logic required to implement resource span checking.

The DSIRDS macro instruction is used to locate an entry address for the resource in the authorization and resource table, DSIART. DSIRDS might be specified as follows:

```
DSIRDS    SWB=(REG2),LUNAME=LUADDR,ARTPOS=ENTRYADR
```

For this example, the authorization and resource table (ART) entry address for the resource pointed to by LUADDR will be returned in ENTRYADR. The resource will be marked as active.

Figure 5 on page 27 shows the relationships between the operator identification table (OIT), the span name table (SNT), and ART. The relative position of an entry in the operator identification table is represented by the bit position of each entry in the span name table (n bits). The relative position of an entry in the span name table is represented by the bit position of each entry in the authorization and resource table (m bits).

For example, if a user wishes to find whether a particular operator is authorized to issue commands for a particular resource, follow this procedure:

- Use the DSIOIS macro instruction to find the position of the operator's identification in the OIT table. The identification is put in the fullword area pointed to by the OPID operand of the macro instruction. The relative position is returned to the fullword area pointed to by the OITPOS operand.

- Use the DSISSS macro instruction to search the SNT for the bit position that corresponds to the location of the operator identification entry in the OIT. The bit position is specified by the OITPOS operand of the macro instruction. It is best to begin the search at the beginning of the span name table. The SNT address (MVTSNT) is found in the NetView main vector table (MVT). Refer to "MVT — Main Vector Table" on page 142 in this book. The address of the first span entry that corresponds with a bit set to 1 is returned to the fullword area specified by the SNTADDR operand of the macro instruction. Because it is the address of the entry and not its relative position that is returned, the starting address should be stored in another area to be used in any calculations that may be required to establish the entry's position.
- Create a mask byte to check the bit position of the authorization and resource table (ART) that corresponds to the span name table entry position.
- Use the DSIRDS macro instruction to find the address of the specific entry for the resource. The address of the entry is returned to a fullword area specified by the ARTPOS operand of the macro instruction. The resource name is specified on the LUNAME operand.
- Use the mask byte to check whether the corresponding bit is set to 1.

In the example shown in Figure 5, the DSIOIS macro can be used to determine the position of the identification OPID2 in OIT. Position 2 is returned to the area specified by OITPOS. DSISSS can then be used to check bit position 2 in SNT. The first span name with 1 in that position is SPAN2. The address of that entry is returned to the area specified by SNTADDR. Using the starting address and the address returned, and dividing the difference by the length of the SNT entries (found (MVTSNTLN found in MVT), the relative position of SPAN2 can be calculated. A mask byte can then be prepared to test the bit position corresponding to SPAN2 in ART. The DSIRDS macro instruction can then be used to find the address of the resource name in ART. If LU2 is specified in the area pointed to by the LUNAME operand, it is the second entry in ART. The mask byte can then be used at that location, showing that the operator whose identification is OPID2 can issue commands for LU2. If a match is not found, DSISSS can be invoked again to find another span. The starting address specified for the SNTADDR operand should be the address of the entry immediately following SPAN2. This process can be repeated until a span is found or the end of the table is reached.

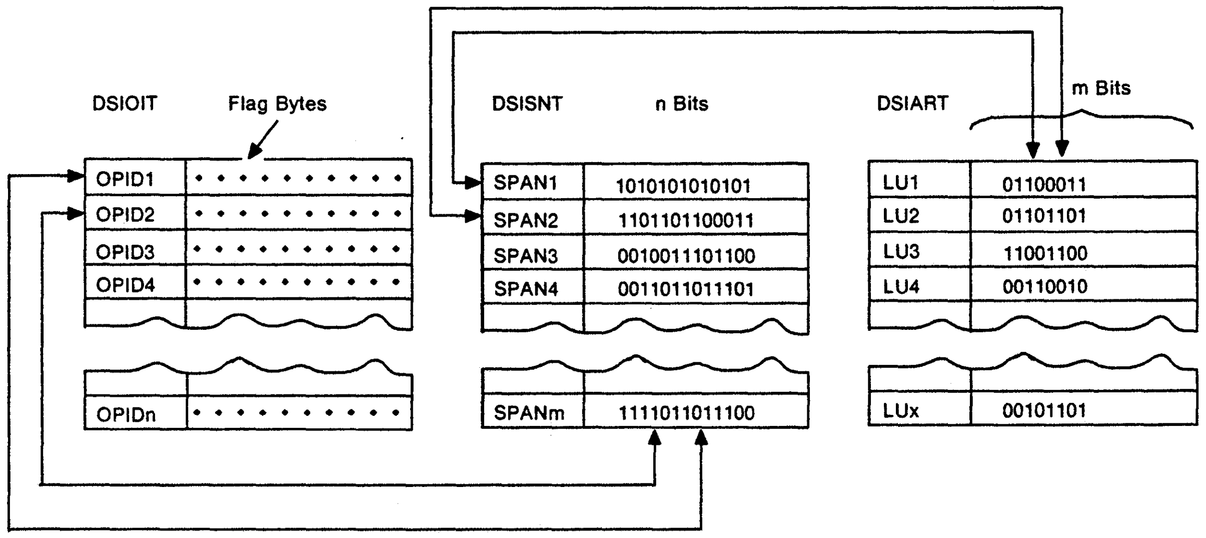


Figure 5. Table Field Relationships

DST (Data Service Task) Unique Services

Access to CNM (Communication Network Management) data and VSAM files is provided by the DSIZCSMS (CNM Service) and DSIZVSMS (VSAM Service) macros. These macro services can only be used under DSTs. DSTs and their macro services are described fully in Chapter 8 on page 161.

Chapter 3

Chapter 3. Writing User Exit Routines

This chapter illustrates how to design, code, and install user exit routines that take advantage of the NetView program's exits at strategic processing points.

Overview of User Exit Routines

You can write user exit routines to view, delete, or replace data flowing to, from, or through NetView. For example, your code can examine the messages passing through NetView, record relevant data, and initiate work requests based on the data. In addition, your code can delete any unnecessary message from further processing or substitute a modified message in place of the original message. Thus, user exit routines handle a specific event with non-standard processing and automate processes based on message information.

NetView provides two types of user exits for which you may write routines:

- Global user exits (DSIEXnn), which apply to all NetView tasks. The global user exit routines are loaded when NetView starts. See Table 2 on page 32 for a list of user exits.
- DST user exits (XITnn and BNJPALEX), which apply only to DST subtasks. (BNJPALEX is MVS only.) The DST user exit routines are loaded when the associated DST starts. Each DST can have its own set of user exit routines. The BNJPALEX exit routine runs under the BNJDSE36 DST.

Note: DST user exits should not be used under the Network Product Support (NPS) task named DSIGDS.

You should avoid coding user exits for frequently executed functions, such as VSAM I/O, since performance can be degraded significantly.

Each user exit handles a particular event, such as the reception of data from the system console. When that event occurs, NetView passes control to the appropriate user exit routine for processing. After processing, the user exit routine returns control and passes a return code to NetView. Optionally, up to 10 DST exit routines can be concatenated. In this case, the first DST exit routine returns control to NetView. If the first exit did not indicate USERDROP, NetView then calls the next one in the sequence. This process continues until the last DST exit has returned control to NetView or USERDROP is indicated.

You do not need to write a routine for each user exit. See "Unused User Exits" on page 48 for more information.

Table 2. User Exit Environments

Exit	Description	Applicable Tasks	TVBINXIT	REENTRANT
DSIEX01*	Input from the Operator	OST with VTAM terminal	On	Yes
DSIEX02	Obsolete; replaced by DSIEX02A			
DSIEX02A	Message Output this Domain or Message Output Cross-Domain	NNT, OST, PPT NNT, OST, CNMCSSIR	On or Off	Yes
DSIEX03	Input Before Command Processing HDRTYPEX Cross-Domain Return Command Receive	NNT, OST, PPT NNT	Off	Yes
DSIEX04	Log Output for Buffers not Processed by DSIEX02A	Main Task or Any Subtask	On or Off	Yes
DSIEX05	Before VTAM Command Invocation**	NNT, OST, PPT	Off	Yes
DSIEX06	Solicited VTAM Messages**	NNT, OST, PPT	Off	Yes
DSIEX07	Cross-Domain Command Send	NNT, OST	Off	Yes
DSIEX08	Obsolete			
DSIEX09	Output to the System Console	Main Task or Any Subtask	On or Off	Yes
DSIEX10	Input from the System Console	Main Task	Off	No
DSIEX11	Unsolicited VTAM Messages**	PPT	Off	No
DSIEX12	Logon Validation	NNT, OST	Off	Yes
DSIEX13	OST/NNT Message Receiver	NNT, OST, PPT	Off	Yes
DSIEX14	Before Logoff	NNT, OST	Off	Yes
DSIEX16	Post-Message Automation Table Exit	NNT, OST,PPT CNMCSSIR	On	Yes
BNJPALEX	Screen 4700 loop alerts	DST (BNJDSE36)	Off	Yes
XITBN	BSAM Empty File	DST	Off	No***
XITBO	BSAM Output	DST	Off	No***
XITCI	CNM Interface Input	DST	Off	No***
XITCO	CNM Interface Output	DST	Off	No***
XITDI	DST Initialization	DST	Off	No***
XITVI	VSAM Input	DST	Off	No***
XITVN	VSAM Empty File	DST	Off	No***
XITVO	VSAM Output	DST	Off	No***
XITXL	External Logging	DST	Off	No***

Note:

*	Does not apply to AUTOTASK and MVS console task.
**	When using NetView POI only. Does not include messages from MVS/XA SSI. You can process these messages in DSIEX02A.
***	If used by more than one DST, then they must be reentrant.

Designing and Coding a User Exit Routine

User exit routines must adhere to the guidelines for user-written programming described in "General Coding Guidelines" on page 8. In addition, user exit routines must conform to the special requirements described in this section. After coding your user exit routine, follow the instructions under "Installing a User Exit Routine" on page 48.

Input

Upon entry to the user exit routine, the registers contain the following information:

Register	Contents
1	The address of the user exit parameter list (USE). The USE contains the following: • The address of a service work block (SWB) to be used as a work area or to request services from NetView (USERSWB). If you use the SWB for your save area or for dynamic variables, you must obtain another SWB when invoking NetView macros. • The address of the message buffer (USERMSG) • The address of the TVB.
13	The address of a standard 72-byte save area used to store the caller's registers.
14	The return address.
15	The entry address of the user exit.
0, 2 - 12	Unspecified.

Output

User exit routines pass the following return codes to NetView in register 15 to indicate that the messages are to be unchanged, deleted, or replaced:

Return Code	Meaning
USERASIS (0)	Use the message as presented to the user exit; do not delete or replace it.
USERDROP (4)	Delete the message from the terminal and from the network log, system log, and hard copylog; stop processing before the message appears on the screen. For more information on how to delete messages, see "Deleting Messages" on page 34.
USERSWAP (8)	Replace the message with the message addressed in register 0. For more information on how to replace messages, see "Replacing Messages" on page 34.

Do not change any input, particularly the USERMSG buffer in the USE control block. Aside from register 15 (and register 0 if USERSWAP is returned), the other registers should be restored without change. See the specific user exit descriptions under "Summary of User Exits" on page 36 for additional return code considerations.

Deleting Messages

If you want a message logged but not displayed, you can set the appropriate display and logging flags in the IFRCODAI (see "IFR — Internal Function Request" on page 133) in user exit DSIEX02A.

To delete a message entirely, use return code USERDROP. NetView will free the message buffers using DSIFRE. Therefore, the user exit should not free the buffers.

When NetView receives a USERDROP return code, no further exit routines are called. Thus, if you have concatenated DST exit routines, a USERDROP return code prevents the next exit routine from being called.

When processing a single line of a title-line message, (HDRTYPEJ, HDRTYPEK, or HDRTYPEL) **do not delete** a CONTROL, LABEL, or END line unless the entire message is deleted. When processing a title-line message formatted as an IFRCODAI, you may delete any line. For example, if DSIEX06 deletes message IST314I END (only), processing of the entire title-line message (of which this is the END line) would be disrupted. However, if DSIEX02A deletes IST314I END, then the remainder of the title-line message would be displayed normally.

Replacing Messages

When replacing a message, the new message must either be a static message or be in a buffer in a reentrant storage area, such as the SWBADATD or SWBPLIST areas of the USE control block's USERSWB. Only the text portion of the buffer is swapped. Also, make sure that HDRMLENG of the new message is less than or equal to HDRMLENG of the original message. Do not replace a message with a new message that is longer, unless the message is formatted as an IFRCODAI.

If you want to replace a title-line message, do not change the HDRMTYPE or HDRIND fields in the buffer header. For more information on title-line messages, see "Title-Line Output" on page 19.

When processing a single line of a title-line message (HDRTYPEJ, HDRTYPEK, or HDRTYPEL), **do not replace any lines** or the sequence and format may be lost. When processing a title-line message formatted as an IFRCODAI, you may replace any line.

User exit DSIEX02A provides a more flexible interface for replacing messages including title-line and MLWTO (multiline write-to-operator) type messages. See "DSIEX02A: Output to the Operator" on page 37.

You can concatenate DST user exit routines when replacing messages. In this case, the buffer containing the replacement message becomes the input for the subsequent DST user exit routine.

Message Flow and Interception Points

The following is the sequence of decisions made in handling a message in NetView. This information may be useful in determining what forms of message processing would be most efficient to meet the performance objectives at your installation.

If a message is suppressed, dropped, or deleted, the message is removed from the flow and does not proceed further.

OST/NNT Task

- If the message is solicited from VTAM via the POI interface:
  - If this is one of the messages that status monitor uses to update network status, it is processed by status monitor.
  - If there is a user exit DSIEX06 (solicited access method message input), the exit is invoked. Deleted messages are not processed further.
- Exit DSIEX02A is called (if one exists). If the exit indicates that the message is to be dropped, it is deleted.
- The message is checked to see if it satisfies an &WAIT condition in a NetView command list or a TRAP condition in a REXX or high-level language command procedure. With the SUPPRESS option, the message is marked for deletion unless DSIEX16 specifies otherwise.
- Message automation table processing begins. Table actions are reflected in the buffer structure given to DSIEX16.
- DSIEX16 is called.

  The message automation table and DSIEX16 are called only once for each unique message in a NetView domain. Any copies of the message made by the ASSIGN command or the message automation table will not result in a call to the message automation table or DSIEX16 in this NetView domain.
- Logging, display, routing, and command actions are processed as specified in the buffer in combination with the current DEFAULTS and OVERRIDE settings.
- If the message is displayed and copies have been assigned by the ASSIGN command with the COPY option, a copy is sent to each assigned operator. The copied messages cannot be automated by the message automation table.

PPT Task Including Messages to the Authorized Receiver

- If the message is from VTAM via the POI interface:
  - If this is one of the messages that the status monitor uses to update network status, it is processed by the status monitor.
  - If there is a user exit DSIEX11 (unsolicited message) or DSIEX06 (message solicited by a VTAM command issued under the PPT task), the proper exit is invoked. Deleted messages are not processed further.
  - VTAM commands and messages received due to the VTAM start option PPOLOG=YES (such as those entered from the system console and merely echoed to NetView) are logged only. They will not be automated.
- If the message has been assigned using the ASSIGN command with the PRI and SEC options, each assigned operator will be sent a copy of the message. These messages then proceed through the OST/NNT flow for those particular operators, but the secondary (SEC) copies will not be processed by the message automation table or DSIEX16.
- If the message has not been assigned, it is sent to the authorized message receiver and proceeds through the OST/NNT flow for that operator. See the *NetView Administration Reference* for information on how the authorized message receiver is determined.
- If the message has not been assigned and no authorized message receiver is logged on to NetView, the flow continues as follows:

- Exit DSIEX02A is called (if one exists). If the exit indicates that the message is to be dropped, it is deleted.
- Message automation table processing begins. Table actions are reflected in the buffer structure given to DSIEX16.
- DSIEX16 is called.

  The message automation table and DSIEX16 are called only once for each unique message in a NetView domain. Any copies of the message made by the ASSIGN command or the message automation table will not result in a call to the message automation table or DSIEX16 in this NetView domain.
- Logging, routing, and command actions are processed as specified in the buffer in combination with the current DEFAULTS and OVERRIDE settings.
- If the message is to be displayed, it is written to the system console.

Control Blocks

The service facilities used in your user exit routine always dictate which control blocks you must include in your routine. However, you will always need these three control blocks:

- The user exit parameter list (USE)
- The main vector table (MVT)
- The service work block (SWB).

Use macro DSICBS to include these and any additional control blocks your routine needs. For details, see "DSICBS – Control Block Services" on page 162 and the control block descriptions in Chapter 7 on page 119.

Summary of User Exits

This section describes each user exit that NetView provides, including examples of use and coding requirements.

DSIEX01: Input from the Operator

Description: NetView calls DSIEX01 when an operator provides standard-mode input to an OST or when an NNT receives cross-domain input. DSIEX01 runs after device-dependent data has been removed from the input buffer but before syntax or command verbs are analyzed and before the message is logged. All commands issued from the command facility, hardware monitor, or the threshold analysis and remote access feature are passed to exit 01. (Hardware monitor and the threshold analysis and remote access feature commands have a prefix: "CMD==> or CMD=>"). All regular commands, including those from the command facility, hardware monitor, or the threshold analysis and remote access feature are passed to exit 03.

Example of Use: You can use DSIEX01 to count the times an immediate command has been called.

Coding Considerations: DSIEX01 follows the coding guidelines given under "Restrictions when TVBINXIT is On" on page 9. DSIEX01 does not apply to unattended operator tasks and MVS console operator tasks.

DSIEX02: Output to the Operator

Description: This exit is obsolete.

DSIEX02A: Output to the Operator

Description: NetView calls DSIEX02A for standard output to an operator's terminal (DSIPSS TYPE=OUTPUT, DSIPSS TYPE=IMMED, or FLASH). DSIEX02A runs before the device-dependent output is inserted and before the data is logged.

The TVBINXIT bit in DSITVB indicates the environment in which the user exit routine is running. For more information on TVBINXIT, see page 8.

Example of Use: Since the message has been formatted but not yet displayed or logged, you can use DSIEX02A to delete or replace the message before it is automated, logged, or displayed.

If your messages will be translated (such as to Kanji), changes to the message text may affect the translations. (See *NetView Installation and Administration Guide* for more information.)

Coding Considerations: If TVBINXIT is on, DSIEX02A follows the coding guidelines given under "Restrictions when TVBINXIT is On" on page 9.

Do not use macro DSIPSS in this user exit routine. If a message is required, use DSIMQS to queue the message to the subtask.

Since NetView does not check the syntax of messages that are sent to a terminal, DSIEX02A does not receive a parse descriptor block (PDB).

Message automation is invoked after this exit routine has been called; therefore, any changes made for messages in this user exit may affect message automation. Message automation is not invoked for a message that has been deleted by this exit routine.

DSIEX02A provides the following additional features not available in other exits:

- The NetView buffer passed to this exit is an IFRCODAI internal function request. You will need to reference the IFR control block rather than the BUFHDR.
- For single line messages, multiple line messages, and title-line messages, this buffer points to the entire chain of buffers that comprise the message.

All chained buffers can be replaced by using DSIGET for non-queued subpool zero storage for new buffers and copying or replacing all the data in the old buffers. Any original buffers passed to the exit should be either freed or passed back to NetView on the return. The unused buffers must be freed using DSIFRE for non-queued subpool zero storage. You must be careful to initialize all necessary fields in all buffers and copy any reserved or unused header information from each of the buffers. (The IFRCODAI buffer should *not* be freed.)

Note: NetView will free all returned buffers that are in the IFRCODAI format.

The IFRCODAI contains control information and MVS system data which should only be accessed through the provided NetView message table and command list interfaces. Unauthorized modification of these fields may cause processing, logging, or display loops.

Under MVS/XA, DSIEX02A is supported only in 31-bit addressing mode.

Return Code Considerations:

Return Code	Meaning
USERDROP	NetView will free all of the buffers passed to the exit. Thus DSIEX02A should not DSIFRE the buffers when using USERDROP. Buffers to be freed by NetView are pointed to by USERMSG.
USERSWAP	NetView requires register 0 to point to the original buffer that was passed to the user exit. If the buffer chain is modified, pointers within the buffers that are used for chaining purposes will also have to be changed (such as IFRAUTBA, IFRAUTBL, and HDRNEXTM). You must free any data buffers you remove from the IFRAUTBA chain. NetView will free all returned buffers based upon the address in register 0, including the IFRCODAI buffer.
USERASIS	NetView uses and frees the original buffers using USERMSG.

Note: USERSWAP is identical in function to USERASIS except that register 0 is used instead of USERMSG to find the buffers.

DSIEX03: Input Before Command Processing

Description: All regular commands call DSIEX03. This type of command includes the following:

- Commands issued by a command procedure
- Commands received from another subtask
- Commands used to start the hard-copy log at logon
- Commands used as the initial command
- Commands resulting from the message automation table
- Commands entered for an MVS console operator task
- Commands entered from a terminal
- Commands received as HDRTYPEX messages from an NNT
- Commands queued with EXCMD.

Before running, all commands are passed to either DSIEX01 or DSIEX03. Immediate commands are passed to DSIEX01. Regular commands entered from a command facility, threshold analysis and remote access feature or hardware monitor screen are passed to DSIEX01 and DSIEX03. The remaining command types previously listed are passed to DSIEX03.

Example of Use: You can use DSIEX03 to restrict usage of particular, regular commands if your conditions are more complex than those provided by scope checking.

Coding Considerations: None.

DSIEX04: Log Output

Description: NetView calls DSIEX04 during the logging and tracing processes. DSIEX04 is located within log services and applies to messages logged on the network log, the external trace log, the MVS system log, and the hard-copy log. It runs before the message is reformatted and sent to the log.

DSIEX04 is not called if DSIEX02A is called since logging options may be specified either in DSIEX02A or in the message automation table. DSIEX04 is called only if the DSIWLS macro is issued from other than a DSIPSS request.

Example of Use: You can use DSIEX04 to edit information sent to the network log, to the MVS system log, or to the hard-copy log. You can send certain messages to a specific log or to no log at all.

Coding Considerations: If TVBINXIT is on, DSIEX04 follows the coding guidelines given under "Restrictions when TVBINXIT is On" on page 9.

Do not use macro DSIWCS or DSIWLS in this user exit routine.

Since DSIEX04 can run under any subtask that issues macro DSIWLS, be sure that any service facilities you request are supported by the subtask under which the routine is running. For example, the DST subtask is restricted to VSAM or CNM interface services.

Since NetView does not check the syntax of messages that are sent to a log, DSIEX04 does not receive a parse descriptor block (PDB). See "PDB — Parse Descriptor Block" on page 146, if you wish to parse the messages in DSIEX04.

Return Code Considerations: DSIEX04 may pass four other return codes in addition to USERASIS, USERDROP, and USERSWAP.

Return Code	Meaning
USERLOG	Write the message to the network or MVS system log only.
USERLOGR	Write the substituted message to the network or MVS system log only. The address of the buffer containing the new message is in register 0.
USERHCL	Write the message to the hard-copy log only.
USERHCLR	Write the substituted message to the hard-copy log only. The address of the buffer containing the new message is in register 0.

DSIEX05: Before VTAM Command Invocation

Description: NetView calls DSIEX05 when preparing to pass a command to VTAM via the POI interface; domain qualifiers have been removed and all span checking has been completed.

Example of Use: You can use DSIEX05 to verify that an operator is authorized to issue a particular command.

Coding Considerations: Code the routine to handle both the OST and PPT control block structures.

This exit applies only to commands entered directly (not using the 'MVS' prefix) that are passed through NetView's POI.

Note: Commands passed to DSIEX05 have already been processed under DSIEX03 (and, perhaps, DSIEX01).

DSIEX06: Solicited VTAM Messages

Description: NetView calls DSIEX06 when it receives a solicited VTAM message via the POI interface, which is generated in response to a VTAM command the user issued or the PPT issued. The message has not yet been processed or logged.

Example of Use: You can use DSIEX06 to change the message number or text of a VTAM message or to process VTAM messages.

Coding Considerations: Code the routine to handle both the OST and PPT control block structures.

This exit applies only to messages received through NetView's POI in response to commands entered directly (not using the 'MVS' prefix).

Message automation is invoked after this exit routine has been called; therefore, any changes made for messages in this user exit may affect message automation. Message automation is not invoked for a message that has been deleted by this exit routine.

Note: Messages processed (and not dropped) by DSIEX06 will subsequently be processed by DSIEX02A.

DSIEX07: Cross-Domain Command Send

Description: NetView calls DSIEX07 before commands are sent cross-domain to an NNT (DSIPSS TYPE=XSEND).

Example of Use: You can use DSIEX07 to monitor cross-domain traffic through the network.

Coding Considerations: Do not use DSIPSS TYPE=XSEND in this user exit routine. Also, avoid directly calling commands that route a command to another domain, such as ROUTE, DISPLAY, or VARY. If necessary, you may queue such commands for execution (see "Simulating Terminal Input" on page 22 and "Building an IFRCODCR" on page 23).

DSIEX07 does not receive a PDB. The cross-domain NetView parses the messages after they are received. See "PDB — Parse Descriptor Block" on page 146, if you wish to parse the messages in DSIEX07.

DSIEX08: Cross-Domain Message Receive

Description: This exit is obsolete.

Migration Considerations: Rewrite code into DSIEX03 and DSIEX02A using HDRDOMID to identify commands and messages received from other domains.

DSIEX09: Output to the System Console

Description: NetView calls DSIEX09 when a message is written to the system console operator using macro DSIWCS. The message has not been formatted for transmission.

Example of Use: You can use DSIEX09 to edit messages sent to the system console.

Coding Considerations: If TVBINXIT is on, DSIEX09 follows the coding guidelines given under "Restrictions when TVBINXIT is On" on page 9.

DSIEX09 is called as a result of DSIWCS macro calls. The output of the MVS console operator task (OST) is processed in DSIEX02A instead of DSIEX09.

Do not use macros DSIWCS or DSIMQS in this user exit routine. If you need to send a message to the system console from this exit routine, use system macros instead.

Since NetView does not check the syntax of messages that are sent to the system console, DSIEX09 does not receive a parse descriptor block (PDB). See "PDB — Parse Descriptor Block" on page 146, if you wish to parse the messages in DSIEX09.

DSIEX10: Input from the System Console

Description: NetView calls DSIEX10 when input is received from the system console operator. The exit is called after the command has been entered but before it is invoked or logged.

Example of Use: You can use DSIEX10 to allow the system console operator to enter command abbreviations and synonyms. These could then be expanded in the user exit routine.

Coding Considerations: DSIEX10 can only be called from the main task, not from a subtask.

DSIEX10 does not receive a parse descriptor block (PDB). See "PDB — Parse Descriptor Block" on page 146, if you wish to parse the messages in DSIEX10.

DSIEX10 is not called for commands entered by an operator using an MVS console operator task (OST). DSIEX03 is called instead.

DSIEX11: Unsolicited VTAM Messages

Description: NetView calls DSIEX11 when an unsolicited VTAM message is received via the POI interface. In addition, when VTAM's PPOLOG=YES modify or start option is used, copies are presented to DSIEX11. This user exit is called before the resource name is analyzed and before the message is logged.

Example of Use: DSIEX11 can issue macro DSIMQS to send a copy of the message buffer prior to processing by NetView. If you want to send unsolicited messages to all operators, DSIEX11 can transform the messages into MSG ALL commands.

Coding Considerations: If DSIEX11 calls a command or a command procedure, the command restrictions for the PPT apply.

Message automation is invoked after this exit routine has been called; therefore, any changes made for messages in this user exit can affect message automation. Message automation will not be invoked for a message that has been deleted by this exit routine.

DSIEX12: Logon Validation

Description: NetView calls DSIEX12 at the completion of the logon process, after the logon has been accepted by NetView.

Example of Use: You can use DSIEX12 to perform additional checking of authorization and environmental customization. DSIEX12 can also send messages to other operators.

Coding Considerations: If output to the screen is required, use only the following DSIPSS TYPES: SCRSIZE, WINDOW, ASYPANEL, CANCEL, PSSWAIT, and TESTWAIT.

Return Code Considerations: If the user exit routine issues a return code of zero, the logon proceeds. If specified, your hard-copy log starts and the initial command runs. If the issued return code is nonzero, the operator is logged off.

This exit is called under all OST and NNT tasks including unattended-operator and MVS- console-operator tasks.

DSIEX13: OST/NNT Message Receiver

Description: NetView calls DSIEX13 when either a message buffer or a user-defined internal function request (IFRCODUS) is received through macro DSIMQS. DSIEX13 is called within the message receiver for subtask-subtask communication. A message buffer is any non-HDRTYPEI (IFR) buffer.

Example of Use: You can use DSIEX13 in conjunction with IFRCODUS to initiate a user function with a buffer. Code DSIEX13 to perform the user function specified by IFRCODUS.

Coding Considerations: None.

Return Code Considerations: When DSIEX13 returns, these buffers are written to the operator terminal with DSIPSS TYPE=OUTPUT, unless return code 4 is issued. The messages are logged after user exit DSIEX02A is called.

DSIEX14: Before Logoff

Description: NetView calls DSIEX14 when an OST or NNT subtask is preparing to end for any of these reasons:

- If LOGOFF is entered at the operator's terminal
- If the subtask LOSTERM exit is driven (VTAM)
- If the subtask is posted to terminate.

The subtask cannot communicate with the operator's terminal at this point. It is possible, however, to issue macro DSIWCS to write to the system console and macro DSIWLS to write entries to the log.

Example of Use: You can use DSIEX14 to save accounting information or update tables.

Coding Considerations: Because there is no buffer associated with logoff processing, DSIEX14 receives neither an input buffer nor a PDB.

Return Code Considerations: NetView ignores any return code received from this user exit routine.

DSIEX16: Post-Message Automation Table Exit

Description: NetView calls DSIEX16 immediately after a message has been considered for automation under the display services (DSIPSS) of NetView. DSIEX16 can be run under the OST, NNT, PPT, or CNMCSSIR task. It receives a description of the total processing to be performed on the message. This exit is called just before logging, display, routing, or command actions are processed. This exit is not called for messages that are deleted by DSIEX02A. However, this exit *is* called for messages that are suppressed by &WAIT, TRAP and SUPPRESS, or the automation table. Message automation table processing occurs before DSIEX16 is called, and the resultant actions are scheduled immediately after this exit.

Example of Use: You can use this exit to modify the processing options for the message and specify or substitute new logging, display, command, or routing actions independently of one another.

This exit can reformat messages by removing buffers, changing buffers, and adding entirely new buffers to the original message. This exit can prevent OVERRIDE command options from taking effect for messages. This exit can also help monitor the effectiveness of message suppression and automation.

Coding Considerations: TVBINXIT will be on when called for DSIPSS TYPE=IMMED. For more information see "Restrictions when TVBINXIT is On" on page 9. Do not use DSIPSS in this exit routine. New messages may be issued by chaining them to the original message structure. DSIEX16 does not receive a parse descriptor block. DSIEX16 uses the IFRCODAI internal function request similar to DSIEX02A.

NetView uses the USERMSG field on return as the chain of IFRCODAI structures to be processed. If USERMSG is set to zero by DSIEX16, the user exit must free all buffers passed. These buffers are non-queued, subpool-zero storage. The buffers are 31-bit mode only for MVS/XA. When USERMSG is non-zero, it must point to the chain of IFRCODAI buffers.

Return Code Considerations: DSIEX16 does not require the use of register 15 return codes. Buffer deletion is indicated by setting USERMSG to zero after freeing any remaining buffers. Buffer substitution is done by manipulating the buffer structures dynamically.

BNJPALEX: Alert Generation Exit Routine

Description: When BNJPALEX is given control, register 1 points to a single element parameter list that points to the following data structure:

AL4	Address of the first extended statistical counter
AL4	Address of an 8-byte user text area
AL4	Address of the first extended statistical counter
AL4	Address of an 8-byte user text area
CL8	Controller name
XL8	Extended statistical counter threshold exceeded bit map (*n*th bit *on* indicates that the *n*th extended statistical counter exceeded the threshold)
XL2	Loop Basic Counter 2 threshold
XL2	Extended statistical counter threshold
XL1	Loop Basic Counter 2 counter value
XL1	Count of extended counters reported (maximum 64)
CL1	Alert type byte **X'01'** Extended statistical counter(s) exceed threshold. **X'02'** Loop Basic counter exceeds threshold. **X'03'** Both counter types exceed thresholds.

When BNJPALEX returns control to the 4700 support facility, register 15 return codes indicate what is to be done with the alert. The following are the only permissible values for register 15:

0 Issue the alert.

4 Issue the alert and include the eight bytes of user text located in the user text area.

8 Do not issue the alert.

Coding Considerations: Each time the 4700 support facility receives data from the 4700 network, it analyzes that data with respect to user-defined error thresholds. Whenever a threshold is exceeded, the 4700 support facility issues an alert message to the NetView authorized receiver. A program exit is provided by the

4700 support facility for screening the loop error alerts (either Loop Basic Counter 2 or extended statistical counter alerts). The optional user-written exit routine may add up to eight bytes of user text in the alert message or it may completely suppress the alert.

If the exit is to be included in the system, it must be linked into a load library specified in the STEPLIB of the NetView start procedure. The name of the load module as well as it's entry point must be BNJPALEX. The exit must be written as a reentrant CSECT.

If BNJPALEX has not been coded, whenever the BNJDSE36 task is started, a message (IEA703I for MVS/XA) will be displayed indicating the LOAD has failed. In this case, the issuance of the IEA703I (for MVS/XA) message is normal.

For an example of an exit routine see *IBM 3600/4700 Threshold Analysis and Remote Access Feature Installation and Customization Guide.*

XITBN: BSAM Empty File

Description: The DST calls XITBN if the DST encounters a BSAM open failure because of an empty data set or file.

Example of Use: You can use XITBN to place a record in the empty data set. You should code this user exit only if you wish to write your own BSAM subtask using DST as a base.

Coding Considerations: XITBN can only use the service facilities available to the DST subtask, excluding macros DSIZVSMS and DSIZCSMS.

Return Code Considerations: To initialize the BSAM data set or file, return the USERSWAP return code and have register 0 point to a buffer that contains the record to be used. A return code other than USERSWAP causes the DST to end.

XITBO: BSAM Output

Description: The DST calls XITBO immediately before the record is written to the BSAM data base.

Example of Use: You can use XITBO to modify the record before it is sent to the BSAM data set or file.

Coding Considerations: XITBO can only use the service facilities available to the DST subtask, excluding macros DSIZVSMS and DSIZCSMS.

You should avoid coding user exits for frequently executed functions, such as BSAM I/O, since performance can be degraded significantly.

XITCI: CNM Interface Input

Description: The DST calls XITCI after CNM data is received.

Example of Use: You can use XITCI to modify CNM interface input data (Deliver RU).

Coding Considerations: XITCI can only use the service facilities available to the DST subtask, excluding macros DSIZVSMS and DSIZCSMS.

Unsolicited cross-domain alerts can cause this user exit to gain control. To determine if the alert came from the local CNMI or the distributed host CNMI, the user exit should check the DSRBCPMS flag. If DSRBCPMS is on, the alert was generated from the distributed host CNMI.

If a substitute buffer is returned in register 0, the data must be a valid SNA request unit (RU). See *Systems Network Architecture Product Formats* for a discussion of RU formats.

XITCI invoked under the DSICRTR subtask allows access to unsolicited CNM data prior to NetView routing (except for cross-domain alerts, which are only accessible under the BNJDSERV subtask). XITCI invoked under any other DST will allow access to CNMI data (both solicited and unsolicited) peculiar to that DST. For example, if an XITCI exit is specified for the BNJDSERV DST, it would be invoked for any solicited or unsolicited data that the hardware monitor processes.

For cross-domain alerts, the DSRBCPMS bit should not be turned off by the user exit. Upon entry to the user exit, HDRTDISP is the offset to the focal point transfer RU Header (NMVT follows the header, for a mapping of this header, please refer to the control block section). HDRMLENG is set to the length of the RU header plus the length of the NMVT. If a substitute buffer is returned, it should be a valid NMVT and only the NMVT data area can be changed. The RU header must remain and the user must copy the RU header into his buffer. The HDRTDISP of the user buffer should be the offset to the RU header and HDRMLENG should be the length of the RU header (44 bytes) plus the length of the new NMVT. The length of the new NMVT cannot exceed the original NMVT, else truncation results. NetView does not recommend substitution of cross-domain alert buffers.

Network Services Request Units are routed as follows:

Request	**Header Value**	**Responsible Subtask Name**
RECMS	X'010381'	BNJDSERV
RECFMS	X'410384'	BNJDSERV and AAUTSKLP
ROUTE-INOP	X'410289'	AAUTSKLP
CNM	X'810814'	AAUTSKLP
NMVT	X'41038D'	As follows

NMVT Request Units are routed based upon the Major Vector Key:

Major Vector Key	**Responsible Subtask Name**
X'0000'	BNJDSERV
X'0001'	BNJDSERV
X'0010'	AAUTSKLP
X'0025'	BNJDSERV
X'006F'	DSIGDS

X'0080'	AAUTSKLP
X'13FF'	BNJDSERV

XITCO: CNM Interface Output

Description: The DST calls XITCO prior to a request for CNM interface output.

Example of Use: You can use XITCO to modify the request for CNM data (Forward RU).

Coding Considerations: XITCO can only use the service facilities available to the DST subtask, excluding macros DSIZVSMS and DSIZCSMS.

If a substitute buffer is returned in register 0, the data must be a valid SNA request unit (RU). See *Systems Network Architecture Technical Overview* for a discussion of RU formats.

XITDI: Data Services Task (DST) Initialization

Description: The DST calls XITDI for each statement read by the DST during initialization. When end-of-file is reached, this user exit is entered and two DSIUSE fields, USERMSG and USERPDB, are set to zero indicating that there is no more data. You can code up to 10 module names for each user-written exit routine. See *NetView Administration Reference* for more information on XITDI during DST initialization. Also see "Data Services Task (DST)" on page 86.

Example of Use: You can add XITDI to the DST initialization deck to provide user initialization values to DST initialization.

Coding Considerations: XITDI can only use the service facilities available to the DST subtask, excluding macros DSIZVSMS and DSIZCSMS. XITDI should not refer to DSRB fields.

Note: If all initialization data is to be processed by XITDI, specify the DST initialization statement that identifies XITDI as the first statement in the DST initialization member.

Return Code Considerations: XITDI can prevent the DST from processing a definition statement by passing return code USERDROP (4) in register 15.

When called for an end-of-file situation, a nonzero return code in register 15 indicates that the DST should be stopped.

XITVI: VSAM Input

Description: The DST calls XITVI after a DSIZVSMS macro is issued. The record has been read from the VSAM data base, but it is not yet passed to the requesting data services command processor.

Example of Use: You can use XITVI to modify the record after it has been retrieved from a VSAM data set or file.

Coding Considerations: XITVI can only use the service facilities available to the DST subtask, excluding macros DSIZVSMS and DSIZCSMS.

You should avoid coding user exits for frequently executed functions, such as VSAM I/O, since performance can be degraded significantly.

XITVN: VSAM Empty File

Description: The DST calls XITVN if the DST encounters a VSAM open failure because of an empty data set or file.

Example of Use: You can use XITVN to place a record in the empty data set. NetView provides its own XITVN for VSAM logs generated under DST. You should code this user exit only if you wish to write your own VSAM subtask using DST as a base.

Coding Considerations: XITVN can only use the service facilities available to the DST subtask, excluding macros DSIZVSMS and DSIZCSMS.

Notes:

1. Only VSAM key-sequenced data sets (KSDS) are supported.
2. Do not replace NetView provided XITVN exits for the DSILOG and DSITRACE subtasks.

Return Code Considerations: To initialize the VSAM data set or file, return the USERSWAP return code and have register 0 point to a buffer that contains the record to be used. A return code other than USERSWAP causes the DST to end.

XITVO: VSAM Output

Description: The DST calls XITVO immediately before the record is written to the VSAM data base.

Example of Use: You can use XITVO to modify the record before it is sent to the VSAM data set or file.

Coding Considerations: XITVO can only use the service facilities available to the DST subtask, excluding macros DSIZVSMS and DSIZCSMS.

You should avoid coding user exits for frequently executed functions, such as VSAM I/O, since performance can be degraded significantly.

XITXL: External Logging

Description: The DST calls XITXL whenever data is to be sent to an external log using DSIWLS with the EXTLOG parameter. For example, session monitor performs external logging of response time and configuration data.

Example of Use: For VM, you can use XITXL to perform the actual logging, since SMF is not available.

Coding Considerations: XITXL can only use the service facilities available to the DST subtask. The buffer passed to the user exit contains the standard header, with HDRTDISP pointing to control block DSIELB. The data that is to be logged follows DSIELB.

Return Code Considerations: For MVS with SMF, the return codes are standard. For MVS without SMF and for all other operating systems, NetView ignores the return code and performs no further processing.

Unused user Exits

NetView attempts to load the global exits (DSIEXnn) and the DST exits (XITnn) specified with DSTINIT statements. If a load attempt fails, NetView issues this message:

```
DSI090I  LOAD FAILED FOR NCCF MODULE exitname
```

If you prefer not to receive this message, you may write a routine for the unused user exit as shown below. However, since null exits slow performance, do not create them (exits DSIEX02A and DSIEX16 are the exception). A null DSIEX02A or DSIEX16 does not degrade performance since no 31- to 24-bit address conversion is done.

```
exitname  CSECT
          SLR   15,15
          BR    14
          END
```

Installing a User Exit Routine

Link-edit the user exit routine load module into the NetView load library. For global user exits, use the appropriate DSIEXnn name. For DST user exits, use a name that you select. Use only one load module for each routine; conditional selection at exit time is not allowed.

Global user exit routines are automatically loaded when NetView starts. DST user exit routines are loaded when the DST starts, provided they have been specified in the DSTINIT statement.

See "Preparing Your Code for Use" on page 4 for information on testing your exit routine before use.

Template for a User Exit Routine

Figure 6 on page 49 shows the basic structure of a user exit routine, including standard entry and exit linkage. This template will run as written for any of the NetView user exits; however, it will perform no functions until you add your code at the designated place in the template. It is available online in the NetView sample library (SYS1.CNMSAMP) as CNMS4282.

```
ATMPUXIT CSECT
***********************************************************************
*                                                                     *
* IEBCOPY   SELECT MEMBER=((CNMS4282,ATMPUXIT,R))                     *
*                                                                     *
* MODULE NAME:                                                        *
*                                                                     *
* FUNCTION:                                                           *
*                                                                     *
*                                                                     *
* INSTALLATION:                                                       *
*                                                                     *
*                                                                     *
* INPUT:   REG 1 - ADDRESS OF USER SERVICE BLOCK (DSIUSE)             *
*          REG13 - ADDRESS OF CALLER'S SAVE AREA                      *
*          REG14 - RETURN ADDRESS                                     *
*          REG15 - ENTRY ADDRESS                                      *
*                                                                     *
* OUTPUT:                                                             *
*                                                                     *
*     REGISTERS:                                                      *
*          REG 0 - WILL CONTAIN ADDRESS OF USER SWAP BUFFER IF        *
*                  USERSWAP RETURN CODE USED, ELSE RESTORED           *
*          REG 1 - REG14 - RESTORED UPON RETURN                       *
*                                                                     *
*          REG 15 RETURN CODES:                                       *
*                  USERASIS (0) - OK, CONTINUE PROCESSING             *
*                  USERDROP (4) - DELETE DATA BUFFER AND END          *
*                                 PROCESSING                          *
*                  USERSWAP (8) - SWAP BUFFER SUPPLIED BY REG 0       *
*                                 AND CONTINUE PROCESSING             *
*                                                                     *
* NETVIEW MACROS:                                                     *
*                                                                     *
*          DSICBS - CONTROL BLOCK SERVICE                             *
*                                                                     *
***********************************************************************
         USING *,15
         B     PROLOG
         DC    C'ATMPUXIT  &SYSDATE. at &SYSTIME.'
PROLOG   DS    0H
         STM   14,12,12(13)        SAVE REGISTERS
         DROP  15
         LR    12,15               SAVE BASE REGISTER
         USING ATMPUXIT,12         REG 12 IS THE BASE
         USING DSIUSE,1            REG 1 POINTS TO DSIUSE
         L     11,USERSWB          LOAD REG 11 WITH SWB ADDRESS
         USING DSISWB,11           BASE SWB
         LA    2,SWBSAVEA          GET ADDRESS IF SAVE AREA
         ST    2,8(,13)            SAVE REG 2
         ST    13,4(,2)            SAVE REG 13
         LR    13,2                REG 13 CONTAINS SAVE AREA ADDR
         LR    9,1                 MOVE DSIUSE ADDRESS
```

Figure 6 (Part 1 of 4). Template for a User Exit Routine

```
         DROP  1                   DROP ORIGINAL BASE
         USING DSIUSE,9            REG 9 POINTS TO DSIUSE
         L     10,USERPDB          LOAD REG 10 WITH PDB ADDR
         USING DSIPDB,10           BASE THE PDB
         L     8,USERTVB           ADDRESS THE TVB
         USING DSITVB,8            BASE THE TVB
         L     7,TVBMVT            GET THE ADDRESS OF THE MVT
         USING DSIMVT,7            BASE THE MVT
***********************************************************************
*                                                                     *
*     NOW OBTAIN ANOTHER SWB IN ORDER TO ISSUE NETVIEW SERVICE MACROS *
*                                                                     *
***********************************************************************
         DSILCS CBADDR=MYSWBPTR,SWB=GET  GET A NEW SWB
         SPACE 1
* NOTE:  SEE DSISWB DSECT AT THE END OF THE LISTING
         SPACE 1
         LTR   15,15               TEST DSILCS RETURN CODE
         BNZ   ASIS                Simply Return
*                                  (Alternatively design and perform
*                                   some user notification)
         L     5,MYSWBPTR          POINT TO NEW SWB
         L     4,TVBTIB            PUT THE TIB ADDRESS IN REG 4
         ST    4,SWBTIB-DSISWB(,5) STORE MY TIB ADDR IN THE NEW SWB
***********************************************************************
*                                                                     *
*   THIS NEW SWB (MYSWBPTR) SHOULD BE USED FOR SERVICE MACROS.        *
*                                                                     *
***********************************************************************
*        PUT YOUR USER EXIT CODE HERE                                 *
*         ...                                                         *
*         ...                                                         *
*         ...                                                         *
*         ...                                                         *
         BAL   14,FREESWB          Free the MYSWBPTR SWB
*         ...                                                         *
*        Branch to return processing you require (ASIS, DROP, or SWAP)*
***********************************************************************
*   PICK THE EXIT LINKAGE DESIRED FROM THE THREE BELOW:
***********************************************************************
*---------------------------------------------------------------------
* ASIS:  TO PROCESS THE BUFFER AS IT IS FROM HERE ON, RETURN FROM HERE
*---------------------------------------------------------------------
ASIS     LA    15,USERASIS         SET AN ASIS RETURN CODE
         L     13,4(,13)           RESTORE CALLER'S SAVE AREA ADDR
         L     14,12(,13)          RESTORE CALLER'S REGISTER 14
         LM    0,12,20(13)         RESTORE CALLER'S REGISTERS 0-12
         BR    14                  RETURN TO CALLER
*---------------------------------------------------------------------
* DROP:  TO STOP FURTHER PROCESSING ON THIS BUFFER, RETURN FROM HERE
*---------------------------------------------------------------------
DROP     LA    15,USERDROP         SET A DROP RETURN CODE
         L     13,4(,13)           RESTORE CALLER'S SAVE AREA ADDR
         L     14,12(,13)          RESTORE CALLER'S REGISTER 14
         LM    0,12,20(13)         RESTORE CALLER'S REGISTERS 0-12
         BR    14                  RETURN TO CALLER
         SPACE 1
```

Figure 6 (Part 2 of 4). Template for a User Exit Routine

```
*---------------------------------------------------------------------
* SWAP:  TO SWAP A BUFFER FOR THE BUFFER PASSED, RETURN FROM HERE
*---------------------------------------------------------------------
SWAP     LA    15,USERSWAP          SET A SWAP RETURN CODE
         L     0,SWAPBFR            POINT TO THE SWAP BUFFER
         L     13,4(,13)            RESTORE CALLER'S SAVE AREA ADDR
         L     14,12(,13)           RESTORE CALLER'S REGISTER 14
         LM    1,12,24(13)          RESTORE CALLER'S REGISTERS 1-12
         BR    14                   RETURN TO CALLER
         SPACE 1
***********************************************************************
* Subroutine: FREESWB
*   Function: Free the SWB addressed by MYSWBPTR
*   Note:  The new SWB must be released before exiting
***********************************************************************
FREESWB  EQU   *
         ST    14,SAVE14            Save caller's return address
         DSILCS CBADDR=MYSWBPTR,SWB=FREE   NOW FREE THE GOTTEN SWB
*                                   No recovery for failure can be made
         L     14,SAVE14
         BR    14                   Return to call point
```

Figure 6 (Part 3 of 4). Template for a User Exit Routine

```
***********************************************************************
*       Declares and DSECTs
***********************************************************************
*       Include the required control blocks
        DSICBS DSITIB,DSITVB,DSIMVT,DSISWB,DSIPDB,DSIUSE,DSISVL,      *
              PRINT=NO            Suppress Control Block Listing
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
DSISWB   DSECT ,                    EXTEND THE SWB DEFINITION
         ORG   SWBADATD             POINT TO 256 BYTE WORK AREA
WORKAREA DS    0CL256               WORKAREA IS 256 BYTES LONG
MYSWBPTR DS    A                    ADDRESS OF MY NEW SWB SAVED HERE
SWAPBFR  DS    A                    ADDRESS OF SUBSTITUTION BUFFER
SAVE14   DS    A                    Save area for Register 14
         SPACE 1
ATMPUXIT CSECT ,                    RESUME CSECT
         END   ATMPUXIT             END OF THE USER EXIT
```

Figure 6 (Part 4 of 4). Template for a User Exit Routine

Chapter 4

Chapter 4. Writing Command Processors in Assembler Language

This chapter illustrates how to design, code and install assembler language command processors for NetView. Command processors perform a particular service or function, such as extracting relevant data from a control block and presenting the data to an operator.

Overview of Command Processors

Command processors run in any of several execution environments, as allowed by the command processor's *type*, defined by the CMDMDL statement in DSICMD. The CMDMDL statement identifies the command processor as one of the following types:

- Regular commands (TYPE=R)
- Immediate commands (TYPE=I)
- Data services commands (TYPE=D)
- Regular and immediate commands, combined (TYPE=B)
- Regular and data services commands, combined (TYPE=RD).

In addition, long running commands are composed of regular (or type RD or type B) commands. Parts of long running commands are coupled by their internal processing.

Regular Command Processors

A regular command runs under the OST, NNT, or PPT subtask environments and receives control with the TVBINXIT bit set off. This means that the processing of a regular command may be interrupted by the NetView program's system and VTAM's exit routines, as well as by immediate command processors.

For specific coding instructions, see "Writing a Full-Screen Command Processor" on page 61 and "Writing a Long Running Command Processor" on page 64.

Immediate Command Processors

An immediate command runs under the OST and NNT subtask environments. Unlike regular 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.

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.

Data Services Command Processors

A data services command processor (DSCP) runs under the DST subtask environment. DSCPs perform CNM data services using macro DSIZCSMS, or VSAM data services using macro DSIZVSMS, or both services. DSCPS are also appropriate for centralized or serialized user-defined functions that do not use CNM or VSAM services. See "Data Services Command Processor" on page 88 for details.

Combination Command Processors

One type of combination command processor runs as a regular or immediate command, depending on its environment. This command processor checks the TVBINXIT bit and processes the command as an immediate command if the bit is on or as a regular command if the bit is off.

Another type of combination command processor runs as a regular or data services command, depending on the task type indicated in the TVB. If the task type is DST, the command runs as a data services command. Otherwise, the command runs as a regular command.

Long Running Command Processors

A long running command is a command processor that allows other processing to continue after a command has begun processing. DSIPUSH provides for continuation by the same or another processor under varying conditions. The caller of the original command may run after that command returns. Other processing, for example messages, may occur between the calls to the various parts of the long running command processor.

Long running commands run under an OST, NNT, PPT, and (limited) DST. They may be invoked directly by operator input, called by a command procedure, or called by another long running command. They return control to NetView after scheduling work but before processing is complete. NetView then processes other work that may be pending. Only long running commands are capable of acting like a NetView *component*, suspending for unrelated operator commands, including ROLL, and resuming, in turn.

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 procedure to proceed in the midst of this retrieval. During this retrieval, the processor's task may continue to receive messages and accept commands.

For specific coding instructions, see "Writing a Long Running Command Processor" on page 64.

Unattended and MVS Console Operator Task Command Considerations

Command processors using DSIPSS TYPE=ASYPANEL or other full-screen functions should test TVBAUTOO. If this bit is 1, full-screen mode is not permitted. TVBAUTOO indicates an UNATTENDED or MVS console operator task.

Designing and Coding a Command Processor

Command processors must adhere to the guidelines for user-written programming described in "General Coding Guidelines" on page 8. In addition, command processors must conform to the special requirements described in this section. After coding your command processor, follow the instructions under "Installing a Command Processor" on page 71.

To allow for error recovery, consider testing the TVBRESET flag set by the RESET command. You can code your command processor to examine this flag regularly and end itself prematurely if the flag is on.

Input

When the command processor gains control, the registers contain the following information:

Register	Contents
0	Undefined for command invocation. Storage address for long running command resume routines.
1	The address of the command work block (CWB). The CWB contains the following: • A user save area (CWBSAVEA), which is the command processor's 72-byte save area • The address of the command buffer (CWBBUF) for a command call. This field is zero for a RESUME, ABEND reinstate, or LOGOFF call. • The address of a service work block (SWB) for calling service facilities (CWBSWB) • The address of a parse descriptor block (CWBPDB) filled out if CWBBUF does not equal zero • A work area (CWBADATD), which is the command processor's 256-byte temporary storage for keeping variables while remaining reentrant.
13	The address of a standard 72-byte save area used to store the caller's registers.
14	The return address.
15	The entry address of the command processor.
2 - 12	Unspecified.

When a command results from the message automation table, TVBAIIFR will contain the address of the message buffer structure that was automated. If TVBAIIFR is zero, the command did not result from an automated message. (See Figure 8 on page 60 for an example of an automation internal function request buffer structure).

Output

When NetView regains control, register 15 should contain a return code and the other registers should be restored to the caller's contents.

Register	Contents
15	A return code
0 - 14	Restored to caller's contents.

If a *regular* command processor is called by a command procedure (NetView command list language, REXX, or high-level language), the return code is made available to the caller (&RETCODE, RC, HLBRC, respectively). NetView makes no other use of the return code.

For an *immediate* command, NetView ignores the return code.

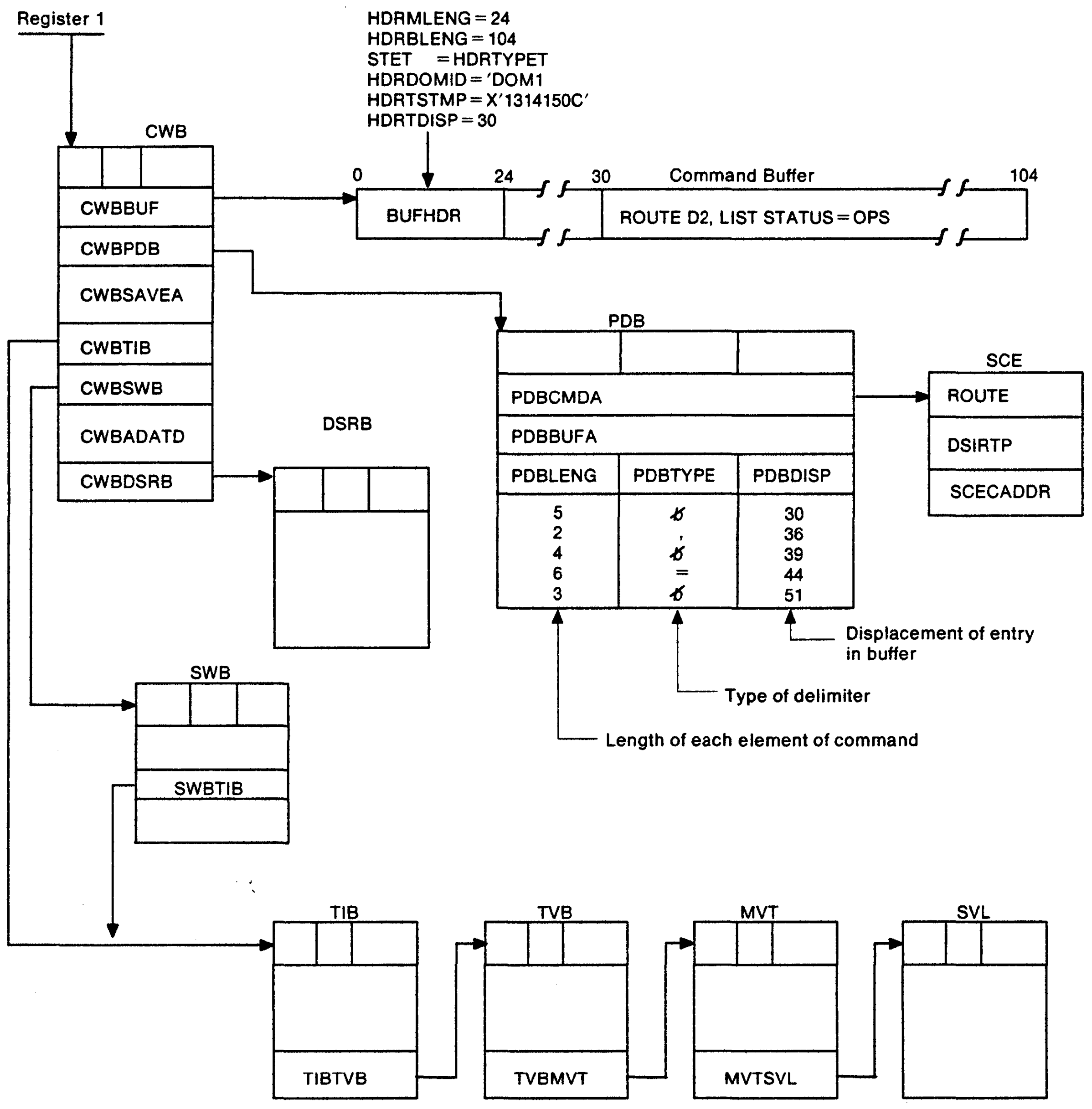
For a *long running* command processor, the completion code is specified on the DSIPOP macro invocation. (See "DSIPOP – Remove Long Running Command" on page 190.) The register 15 return codes returned upon command resumption indicate processing options. (See "Message STIFLE" on page 67.)

Control Blocks

Command processors usually access a command buffer and seven control blocks: CWB, PDB, SWB, TVB, TIB, MVT, and SVL. In addition, type RD command processors running under a DST and type D command processors require the DSRB. Further, a command driven by means of message automation may require the automation IFR.

Figure 7 on page 59 illustrates an example of the required control blocks and their relevant pointers. For detailed descriptions of these control blocks, see Chapter 7 on page 119.

The command buffers CWB, PDB, SWB, DSRB, and AIFR are specific to the command being executed. The TVB and TIB are global to the task, and the MVT and SVL are global to NetView.



NOTE: The NetView control block header (CBH) appears at the beginning of the CWB, DSRB, MVT, PDB, SVL, TIB, and TVB.

Figure 7. Example of Control Blocks Used by Command Processors

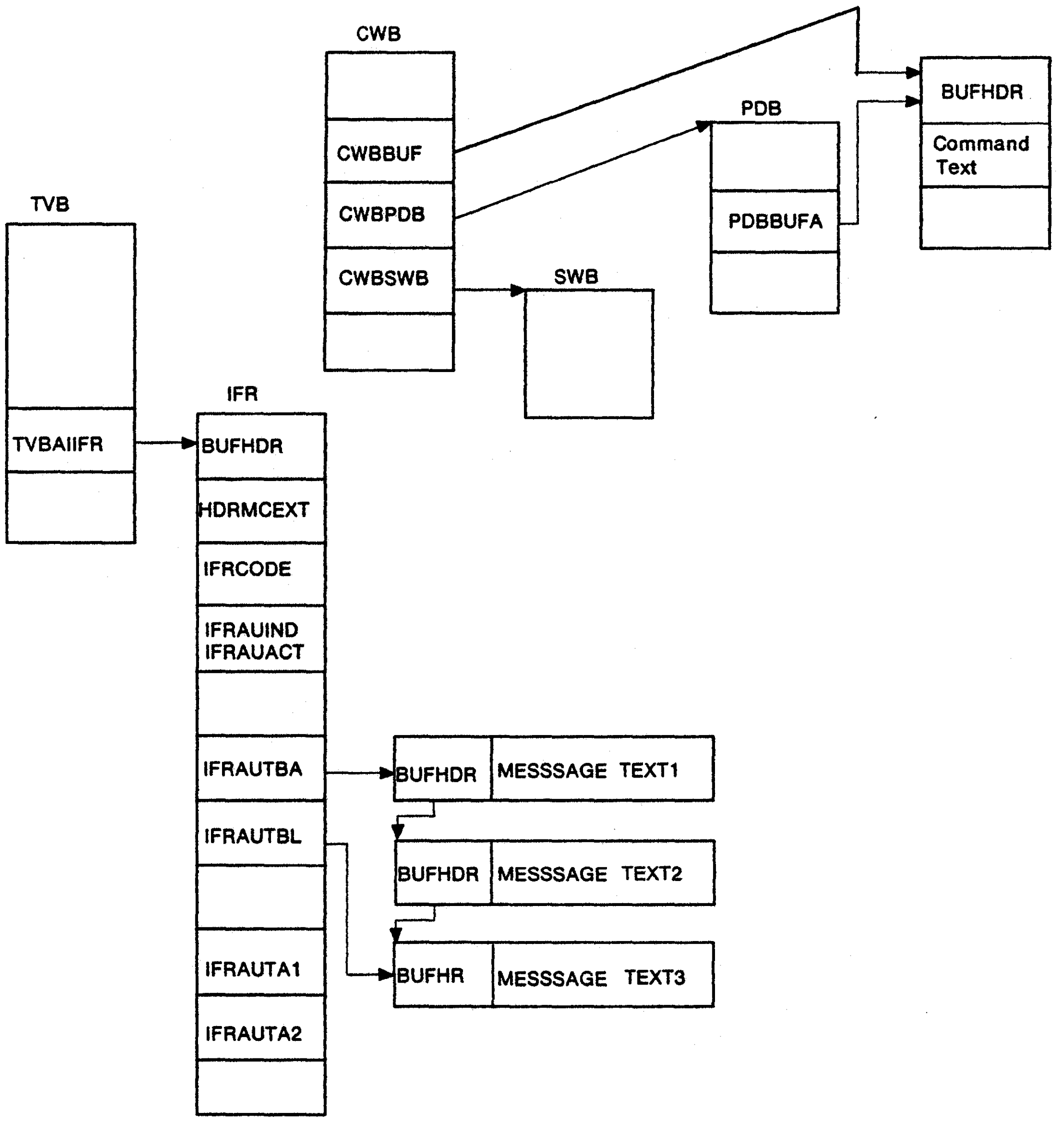


Figure 8. Automation Internal Function Request. An example of a buffer structure when a command processor is driven from an automation statement in the message automation table. If TVBAIIFR=0, this command was not driven by an automation statement.

Figure 9 shows sample code to access a command buffer.

```
         L     R3,CWBPDB              GET ADDRESS OF PDB
         USING DSIPDB,R3              R3 IS BASE FOR PDB
         LA    R4,PDBTABLE            GET ADDRESS OF PDB TABLE
         USING PDBENTRY,R4            R4 IS BASE FOR A PDB ENTRY

* First PDB entry is for command name

         CLC   PDBNOENT,=H'1'         ANY COMMAND PARAMETERS ENTERED?
         BNH   ............           NO, GO HANDLE THIS SITUATION

* Process 1st parameter after command name

         LA    R0,PDBENTND-PDBENTRY   GET LENGTH OF PDB ENTRY
         AR    R4,R0                  BUMP PAST COMMAND NAME ENTRY

         CLI   PDBLENG,0              WAS ONLY A DELIMITER SPECIFIED?
         BE    ............           YES, GO HANDLE THIS SITUATION

         L     R2,CWBBUF              GET ADDRESS OF COMMAND BUFFER
         AH    R2,PDBDISP             POINT TO PARM IN BUFFER
         SLR   R1,R1                  CLEAR LENGTH REGISTER
         IC    R1,PDBLENG             GET LENGTH OF PARM

 ......  application code to process parm ............................

         CLC   PDBNOENT,=H'2'         MORE COMMAND PARAMETERS ENTERED?
         BNH   ............           NO, GO HANDLE THIS SITUATION

* Process 2nd parameter after command name

         LA    R0,PDBENTND-PDBENTRY   GET LENGTH OF PDB ENTRY
         AR    R4,R0                  BUMP TO NEXT ENTRY

         CLI   PDBLENG,0              WAS ONLY A DELIMITER SPECIFIED?
         BE    ............           YES, GO HANDLE THIS SITUATION

         L     R2,CWBBUF              GET ADDRESS OF COMMAND BUFFER
         AH    R2,PDBDISP             POINT TO THIS PARM IN BUFFER
         SLR   R1,R1                  CLEAR LENGTH REGISTER
         IC    R1,PDBLENG             GET LENGTH OF PARM

 ......  application code to process parm ............................


* Process nth parameter after command name

 etc. etc.
```

Figure 9. Sample Code to Access a Command Buffer

Writing a Full-Screen Command Processor

This section describes how to write a full-screen command processor (FSCP) which is a regular command processor that presents a full screen of data to an operator's terminal and runs only under an OST. For line-by-line presentation to an operator's terminal, see "Title-Line Output" on page 19.

An FSCP utilizes DSIPSS TYPE=ASYPANEL to present data. In conjunction with long running command support, an FSCP can be coded to allow additional OST work requests to be processed without ending the full-screen presentation. An FSCP can issue macro DSIPSS to request input and then perform other work before issuing macro DSIPSS TYPE=PSSWAIT to receive the input. In addition, an FSCP has direct access to operator input and can use macro DSIWAT to synchronize an operator scenario.

The issuer of the DSIPSS TYPE=ASYPANEL request can use DSIPSS TYPE=PSSWAIT to wait on important NetView ECBS such as the termination ECB, the solicited POI ECB, the cross domain ECB, message ECBS, the reset ECB, and the user ASYPANEL ECB. When control is returned from the PSSWAIT, it is usually better to check the NetView ECBS first for a post (your return code will be 56). If you expect the action of your input to be short (for example, QUIT), it is acceptable to check your own ASYPANEL ECB first. The value of the post indicates the status of the DSIPSS TYPE=ASYPANEL request. The post codes can be found on page 201.

Screen Formatting for the 3270 Data Stream

Since the FSCP is responsible for the 3270 data stream, the processor issues DSIPSS with TYPE=SCRSIZE to find the presentation space dimensions. If the result is larger than 24 by 80 characters, the processor may use the 3270 Erase/Write Alternate command. Otherwise, it must use the Erase/Write command. For more information on the 3270 data stream, refer to *IBM 3270 Information Display System Data Stream Programmer's Reference.*

When DSIPSS with TYPE=SCRSIZE is issued for a terminal that uses 14-bit or 16-bit addressing and Query support (indicated in the logmode definition), the returned and actual screen size may be larger than the alternate screen size. If so, when the Erase/Write Alternate command is used to address the parts of the screen outside the alternate screen size, a terminal program check results. To avoid this problem, do either of the following:

- Use a 24 by 80 character screen image data stream and use the Erase/Write command instead of the Erase/Write Alternate command.
- Use a Write Structured Field command to create a partition structured field that controls the buffering in the terminal. For a more detailed explanation, see *IBM 3270 Information Display System Data Stream Programmer's Reference.*

You will not normally need to send READ instructions to the terminal. A READ MODIFIED is set up for you and executed whenever your operator uses an AID key. To receive the data, you must specify an input buffer and an input ECB on a DSIPSS TYPE=ASYPANEL request. When asking for input, be certain that you have unlocked the operator's keyboard (set bit 6 of WCC to '1'B). You may do this with the same DSIPSS invocation that requests the input or with an earlier one. You may also choose to reset the modified data tags. After requesting input, do not request input on a further DSIPSS TYPE=ASYPANEL request until your ECB is posted.

Do not free the storage where your input buffer and ECB reside, until the ECB is posted. When necessary, you can force the ECB to be posted early by issuing DSIPSS TYPE=CANCEL. Be aware that after you issue DSIPSS TYPE=CANCEL, the operator will not be able use his terminal until either another input request is made or a DSIPSS TYPE=OUTPUT restores the command facility screen.

Reshow Option: Usually you will want to be able to suspend and resume full-screen processing by using macros DSIFIND, DSIPOP, and DSIPUSH and by specifying a RESUME routine. This routine can present previously saved screen information.

Logging Full-Screen Input/Output: NetView does not automatically log full-screen input and output. Use macro DSIWLS to log pertinent data.

Escape Option: When you write a panel to the terminal, you must allow for operator response by specifying an ECB address and Read buffer with at least one of your output requests. After you have done this, *all* input from the terminal will come to your program.[2] It is customary to respond to PF3 by terminating your FSCP. If you want your FSCP to make a temporary exit under other conditions, you must have previously prepared for resumption using DSIPUSH.

FSCP Functions: When an FSCP sends a full screen of data to the display terminal, the system reads the 3270 data stream into the buffer area. At this point, the FSCP can write more data to the screen while the operator is viewing or entering data. However, avoid writing over, or erasing, the operator's input area(s).

When the data is read, NetView posts an event control block (ECB). Then the command processor processes the input and, optionally, presents more full-screen panels. While the FSCP has a read outstanding, input to the terminal is treated as input to the command processor (not to NetView).

When the command processor is called, it reads and writes to the terminal using DSIPSS TYPE=ASYPANEL. The PANEL parameter of DSIPSS points to a 20-byte parameter list. See "DSIPSS — Presentation Services" on page 196 for details.

DSIPSS with TYPE=PSSWAIT allows the FSCP to wait for both its own list of events and a list of events, such as important messages, *for which the FSCP may choose to interrupt its own processing*. After waiting, the command processor tests the return code to determine if its ECB was posted or if a NetView ECB was posted. If the return code shows that a NetView event was completed, the command may return to NetView to allow the processing of the event to occur. If the panel ECB is posted, the FSCP processes the input in the buffer. In this manner, the command processor has complete control of the screen format. The command processor returns to NetView after saving the screen status to enable future processing. The FSCP can specify a RESUME routine (using DSIPUSH) to enable the full-screen presentation to resume.

DSIPSS with TYPE=TESTWAIT allows the command processor to test whether a NetView event has already been posted. You can use this option before issuing DSIPSS TYPE=ASYPANEL to avoid performing input or output when NetView is already posted. This option allows early detection of interruptions. It also lets you return to NetView with a minimum of screen interruptions.

You do not need to use TYPE=PSSWAIT if you do not wish to allow ″interruptions″, such as messages and their resulting automation, or cross domain commands to process. The command processor can wait on its own list of ECBs. Even if you choose not to wait on other NetView events, you are strongly urged to include the OST termination ECB in the list. This ECB is located in the TVBTECB field of control block TVB. TVBTECB enables the command processor to be aware of any major condition requiring the command processor to clean up and exit. Use only the ECBLIST parameter with TYPE=PSSWAIT in DSIPSS.

DSIPSS with TYPE=CANCEL allows you to change characteristics of the FSCP. These characteristics include the input area length and the ECB address. TYPE=CANCEL can

[2] NetView treats power off/power on and the attention signal as error conditions. For power off/power on, your ECB will be posted for a permanent error and your OST will be placed in termination status. The signal that results from the attention key causes NetView to set TVBRESET. (Your PSSWAIT will also end).

be issued when a DSIPSS TYPE=ASYPANEL is active or inactive. It can also be issued if input from TYPE=ASYPANEL has been posted as complete or is not yet complete. This is sometimes necessary since there is no way to guarantee that the operator will enter data on any given panel. If an active ASYPANEL input request is canceled, the system posts the ECB with a special post code. See page 201 for ECB post codes. The storage where the ASYPANEL ECB is located must not be freed until a DSIPSS TYPE=CANCEL is issued or the ASYPANEL ECB has been posted by NetView for successful or unsuccessful input.

Writing a Long Running Command Processor

Long running command processors use macros DSIPUSH, DSIFIND, and DSIPOP to synchronize the order in which functions are processed so that asynchronous events run in sequence or in parallel. Essentially, a long running command processor synchronizes functions so that programs it initiates complete *before* it ends and so that its callers do not resume until *after* it ends.

DSIPUSH identifies each of three routines that provide for command resumption, as well as recovery and termination. These routines are the RESUME routine, the ABEND reinstate routine, and the LOGOFF routine. DSIFIND locates the storage you associated with the DSIPUSH input name. DSIPOP indicates that a long running command has completed.

When one of these routines receives control, CWBBUF is set to zero and register 0 contains the storage pointer associated with the long running command (0 if no storage). Additionally one of the following is set:

- For a RESUME routine, the TVBRESUM bit is set on.
- For an ABEND reinstate routine, the TVBABEND bit is set on.
- For a LOGOFF routine, the TVBLOGOF bit is set on.

Note: The flags TVBRESUM, TVBABEND, and TVBLOGOF are meaningful only when your input CWBBUF address is zero.

The other registers contain the information described under "Input" on page 57.

When a RESUME, ABEND reinstate, or LOGOFF routine returns control to NetView, all registers must be restored. You must set register 15 to tell NetView what action to take, as described in the following sections.

RESUME Routines

Before invoking any subordinate command processors or command lists, and before issuing DSIPSS TYPE=PSSWAIT (or TESTWAIT), the long running command (LRC) processor should schedule a RESUME routine (using DSIPUSH). The RESUME routine suspends any other active long running command processors and enables the long running command processor to regain control.

The first request at the top of the long running command chain defines the controlling RESUME routine. If you use DSIPUSH while another RESUME routine is in control, the new RESUME routine becomes the controlling routine. All other RESUME routines are temporarily suspended. The period of suspension is dependent on the environment from which the LRC received control. If the LRC was called due to an asynchronous event, such as an operator command or the automation of a message, then the NetView ROLL command (if issued) could move (rotate) the LRC to the bottom of the long running command chain, thereby giving control to the next LRC. If the LRC received control by direct call from another LRC (including direct commands from NetView command list language, REXX, and high-level language command procedures), then the two commands are regarded as being related by that direct call. The ROLL command (if issued) acts against both (or all) such related commands as a *group*, moving them all together and preserving their order. In either case, DSIPOP can remove the topmost RESUME routine, giving control to the next long running command.

Note: Neither the ROLL command nor DSIPOP cause an asychronous interrupt. A command gives up control only by returning to its caller, except for the action of immediate commands.

DSIPOP can also be used to remove (by name) a resume routine that is not at the top of the stack of the long running command chain (not *in control*) when you are resumed. This action is regarded as a *cancelation* of that LRC unless your DSIPOP invocation specifies COMPCDE. If the LRC that was removed was part of a larger group, the calling LRC will be given control as soon as the current process allows and will be given a cancel indication, as follows:

- NetView command list language command lists are stopped
- REXX command lists receive a HALT
- all others receive a -5 completion code.

See *NetView Customization: Using PL/1 and C* for more information on cancelable and non-cancelable high-level language procedures.

All command procedures are long running commands. A command procudure's RESUME routine blocks RESUME routines pushed earlier exactly like other long running command processors. A PAUSE or WAIT state for a command procedure is no exception.

DSIPUSH for a RESUME routine is either major or minor. A major DSIPUSH places the new RESUME routine at the top of the long running command processor stack suspending previously-issued long running commands from executing. A minor DSIPUSH places the new RESUME routine on the stack *after* any leading command procedures (in the same group), and thus allows the leading command procedures to complete before the new RESUME routine gains control. Command procedures already suspended by other long running command processors are not affected.

A major DSIPUSH would be used to suspend a command procedure's execution until your long running command processor executes a DSIPOP. A minor DSIPUSH is used to allow a calling command procedure to complete after which your RESUME routine gains control.

Completion Codes An LRC may return a completion code to the LRC that invoked it (in the same group) by specifing a value for the COMPCDE keyword on the DSIPOP macro. If an LRC was invoked asynchronously, the value specified for COMPCDE is ignored. The completion code is passed to the calling LRC in CWBRCODE, upon resumption.

Return Codes A RESUME routine may return control to its caller many times before it completes, to allow messages, queued commands, called LRC's, and other asychronous work to process. Upon the initial return (when *commanded*, CWBBUF¬=0), the value in register 15 is ignored. After each resumption, the value in register 15 conveys the LRC requirments for STIFLE (For an explanation of STIFLE see "Message STIFLE" on page 67). Register 15 = -8 requests stifle; register 15 = +8 requests no stifle. Meanings for other return codes are reserved. (For compatibility with prior releases, a zero return code is allowed after DSIPOP is issued to remove the LRC from the stack.)

An important part of any RESUME routine's function is screen control. Since the state of the operator's terminal is not known (see "Screen Identifier" on page 69) on entry, the RESUME routine must ensure the operator is not locked out by a panel left over from a previous long running command processor. This may mean issuing a message (DSIPSS TYPE=FLASH) that guarantees that the command facility panel and command line are available to the operator. It might also mean displaying a full-screen panel (DSIPSS TYPE=ASYPANEL).

Note: The screen control requirement means that the NetView-supplied routine DSILRCR8 should *not* be used as a RESUME routine with NetView as was sometimes appropriate with NCCF. DSILRCR8 can be used as an ABEND reinstate or LOGOFF routine if no clean up besides the DSIPOP is needed.

Caution: Care should be taken when using messages for screen control. Issuing a message on every resumption is excessive and can cause looping if the message is automated or routed. Use the screen serial number (TIBSCRSN) to determine when to issue messages upon resumption. Be especially cautious when issuing messages upon resumptions under the PPT task, since all messages are routed and may generate new activity (timer requests, for example) under the PPT.: To assist RESUME routines that display panels, NetView provides status information via TIBSCRID (see page 69). To assist RESUME routines that do not display panels, NetView provides a flag, TIBLRCNP (long running command new promotion), that is set *after* any RESUME routine is removed from the top of the stack (whether via DSIPOP or by use of the ROLL command). By examining this bit, the RESUME routine can determine whether any other long running command processor has executed since it last had control. For example: IF TIBLRCNP='1'B, THEN exercise screen control, ELSE continue.

A completed RESUME routine (one that has issued DSIPOP against itself) need not be concerned with screen control since the following RESUME routine will assume responsibility. Ending messages (which were recommended with NCCF) issued when a long running command processor finished are not appropriate in NetView.

NetView attempts to give the operator an opportunity to recover from operator errors or certain program errors through the use of the attention signal or RESET (NORMAL) command. When attention is signalled or the RESET command executes, a flag, TVBRESET is set. Additionally, an ECB, TVBRESTE, is posted. It is recommended that all commands, and especially long running command processors, test TVBRESET regularly. Whenever it is set, the command should terminate its processing (DSIPOP if appropriate) and return to NetView.

The following scenario illustrates the use of DSIPUSH and DSIPOP:

1. A command list invokes a command, and to complete the request the command processor must request data from a DST.
2. The command processor issues DSIPUSH specifying a RESUME routine. At this point the command processor has become an LRC (long running command).
3. The command processor uses DSIMQS to queue a buffer with IFRCODE set to IFRCODCR containing its request for data to the DST.
4. The command processor returns to its caller. The terminal has been left in whatever state it was in when the command list was running. In this case, we will assume the command facility screen is displayed.
5. This operator's task is idle (the command list is suspended by the DSIPUSH); therefore, the RESUME routine defined earlier by DSIPUSH is immediately called.

6. The command processor finds CWBBUF = 0 (no command is being passed) and TVBRESUM is set, but TIBLRCNP is not set. The command processor returns control to its caller.
7. A message is received and displayed. The command processor is called again as a RESUME routine as in the previous step.
8. The operator issues a full-screen command, which issues its own DSIPUSH and waits for input.
9. The operator exits (or ROLLs away from) this latter command processor.
10. The panel of the second command processor is left in place, and the original command processor's RESUME routine is called. This time TIBLRCNP is set. The command processor issues a FLASH message: "STILL WAITING FOR DATA." The command facility screen is restored by DSIPSS.
11. An IFRCODCR buffer containing the DST reply is received. After issuing DSIFIND, the IFRCODCR command processor places data into the long running command processor's LRC storage (as defined in the original DSIPUSH). The IFRCODCR command processor only saves the data since another LRC may have been in control at this point.
12. The command processor is resumed again. Finding its data request satisfied, it completes its function and issues DSIPOP against itself, using the COMPCDE keyword on DSIPOP to indicate the nature of the completion. The command processor returns a return code of 8 in register 15.

    Note: For compatibility with prior releases, a zero return code is allowed after DSIPOP is issued to remove the LRC from the stack.
13. NetView calls the next RESUME routine, the command list invoking the original command, which then continues.
14. The command list receives the return code (&RETCODE) that was specified for COMPCDE on the DSIPOP.

In some cases a RESUME routine may wish to return control to NetView without giving up control of the operator's display. For example, the screen might be dynamically updated based on information sent by another task in the form of an IFRCODCR message. (See "IFR — Internal Function Request" on page 133.) To assist with such a function, NetView provides two tools: message STIFLE and a screen identifier.

Message STIFLE

A RESUME routine can request a message STIFLE when it returns control to NetView. This means that ordinary line mode messages will not be displayed and the operator's screen is not disturbed by the processing of the messages.

Some messages are displayed by NetView whether or not STIFLE is in effect. These messages are said to *break* STIFLE mode. The following messages can break the STIFLE mode:

1. Action messages with ISTnnnA or DSInnnA identifiers
2. Messages which request a reply (HDRTYPEY, except the ASSIGN=COPY of HDRTYPEY, which does not break STIFLE)
3. Any message issued with DSIPSS TYPE=FLASH (the intended use of DSIPSS TYPE=FLASH is for command echoes and screen control messages).

If STIFLE is broken, it remains off until reinvoked by a RESUME routine.

A request for STIFLE can be honored only while the NetView log or the hardcopy log remains active. NetView counts messages that are stifled and displays message DSI593I to remind the operator that he needs to consult the NetView LOG or hardcopy log to see these messages. A request for STIFLE will not be honored while the command facility screen is in place. It is assumed that the long running command processor will gain control of the screen through the use of DSIPSS TYPE=ASYPANEL before requesting STIFLE.

STIFLE affects only line mode messages. A full-screen display is not affected. If, for example, the result of a START DOMAIN command (the logon panel) is delayed long enough for a long running command processor to gain control, the returning logon panel will be displayed without regard to the STIFLE.

ROLL Function

A ROLL group is NetView's way of allowing the operator to switch from one function, such as hardware monitor, to another function, such as session monitor, and return to the place at which the operator left the function the last time. This is similar to window processing in other applications.

A ROLL group is a set of related DSIPUSH macro requests. DSIPUSH begins a new roll group when it is invoked from an asynchronous command environment. Operator commands, commands generated by automation, and commands scheduled via DSIMQS are asynchronous. A command called directly from another LRC is synchronous. The synchronous LRC is added to the roll group started by the asynchronous LRC and blocks it until a DSIPOP is issued against the synchronous LRC. The current ROLL group is defined as the ROLL group that is currently first on the long running command chain.

The ROLL command treats each specified ROLL group as a unit when manipulating the chain. The ROLL command takes the topmost ROLL group and moves it to the bottom of the stack. All elements within the ROLL group maintain their position within the group.

Roll Group Usage: The simplest ROLL group would be a command that invoked DSIPUSH with a RESUME routine. The RESUME routine allows the command processor to respond to a ROLL request. If this command accepted command input from the operator (via DSIPSS TYPE=ASYPANEL), a ROLL command would cause the LRC to move (rotate) to the bottom of the long running command chain. Eventually, when another LRC has been similarly rotated to the bottom of the long running command stack by the ROLL command (or ended with DSIPOP) the RESUME routine would regain control.

You can use multiple DSIPUSH requests (with different resume routines or merely different storage pointers) to implement things such as a hierarchical panel structure. In this situation, rolling away and then back brings the operator back to the same panel last presented, and ending a panel (the operator uses PF3, and the program invokes DSIPOP) redisplays the panel 'above' the current one.

The following scenario describes how a full-screen function would make use of the ROLL capability.

- Issue DSIPUSH for a RESUME routine.
- Provide a line on your panels for NetView command input, using 3270 data stream orders.

- When the operator enters data on the command line, the input data stream will contain orders that identify the area of the screen into which the operator typed.
- When command entry is detected (use DSICES to verify command), build a standard NetView buffer with HDRMTYPE=HDRTYPET and issue the DSIMQS macro to send the buffer to the operator's (own) task, using TVBOPID as the destination of the DSIMQS.

  Note: You must translate the command input to upper case if the language the operator is using has upper and lower case characters.
- Return to NetView to allow the command to be processed.
- NetView will reinvoke your RESUME routine when the command has completed processing *and* when your RESUME routine is the first routine on the stack (and is, therefore, the current ROLL group).

  If the command entered was ROLL, the ROLL command processor will automatically switch your ROLL group to last.

  If the command entered establishes a new ROLL group, your ROLL group is pushed down on the stack and the new group becomes current. When the operator exits from the new ROLL group, your ROLL group is invoked by NetView calling the topmost RESUME routine.
- You should also detect PF6 and PF18 as ROLL. In this case you would build the command buffer, but you must look up the command name for the DSIROLL load module using DSICES and then issue DSIMQS to queue the buffer as described above.
- In order to allow the operator to switch to your function from a different ROLL group directly without ROLL, you must define a command (which could be your function's command name with no operands provided) as a re-show request.

  When your command processor is entered (and there are no operands) with the re-show requested, you would issue DSIPUSH with PROMOTE=YES to move your ROLL group to the head of the stack. You would then proceed by refreshing the screen from the last panel the operator saw.

  Note: DSIPUSH with PROMOTE=YES exchanges the storage address in the DSIPUSH parameter list with the one already associated with the named request, and returns the old value in register 0. Therefore, you may issue DSIFIND to determine the current address and specify it on the PROMOTE=YES request to make sure the address stays the same. If you do not specify the address, the zero value in the parameter list will replace the current value.

Screen Identifier

Requesting STIFLE is not a guarantee that a long running command processor's panel will not be modified; therefore, a means is provided to determine whether such modifications have occurred. After writing the panel to the screen, a long running command processor should save the value of TIBSCRID. Upon regaining control, a RESUME routine can compare the present and saved values of TIBSCRID to determine whether and to some extent, what type of screen modifications have occurred since it last had control.

TIBSCRID, a four-byte field, consists of two subfields:

TIBSCRSN	The low order three bytes, which form a serial number for the screen's contents. This number is incremented whenever anything is sent to the screen that changes what the operator sees.

TIBSCRM	The high order byte which is a state change indicator. A change in TIBSCRM usually means a DSIPSS TYPE= CANCEL has been issued. It may also mean that a lock keyboard or other non-data 3270 command has been sent to the terminal.

When a long running command processor regains control and TIBSCRID is unchanged, it may resume processing as if it had never lost control.

When TIBSCRM (and *not* TIBSCRSN) has been changed, the long running command processor should reestablish its read by issuing DSIPSS TYPE=ASYPANEL to send a write/unlock (X'F182') to the terminal and respecify the ECB, if any, by which the long running command waits for input. This will avoid the necessity to refresh the screen.

When TIBSCRSN has changed, some visible modification to the long running command processor's panel has been made. It will be necessary to rewrite the entire panel.

ABEND Reinstate Routines

An ABEND reinstate routine performs a required recovery action, such as freeing control blocks, after NetView recovers from a task's abnormal ending (ABEND). ABEND routines cannot be used while running under the DST since DST tasks are not reinstated after abnormal endings. While running under the DST, use a LOGOFF routine instead.

The ABEND routine assesses the damage caused by an abnormal end and either keeps or cancels the long running command. (There must be an ABEND reinstate routine for each command in the stack.) With its return codes in register 15, the ABEND routine notifies the task whether the command is to be kept or removed from the queue and freed.

Return Code	**Meaning**
0	Keep the long running command request queued.
8	Remove the currently queued long running command request from the queue.

If the routine keeps a long running command, the RESUME routine runs the first time the task has no other work to perform. All stacked long running command routines are maintained in their current order. ABEND reinstate routines cannot issue DSIPOP or DSIPUSH.

Once a task recovers from an abnormal ending, all ABEND reinstate routines are called, starting with the top one in the stack, which is the most recent. When the ABEND reinstate routine returns to its task, it specifies whether the associated command is to be left on the stack or removed from the stack.

LOGOFF Routines

A LOGOFF routine gives the command processor control before the task ends. The task is not reinstated, and the command processor can perform any final clean-up processing, such as closing a data set or freeing storage. (There must be a LOGOFF routine for each command in the stack.)

A LOGOFF routine is called sequentially for each request on the queue. LOGOFF routines cannot issue DSIPOP or DSIPUSH.

When the command processor returns to the task, requests are taken off the queue and freed.

NetView ignores all return codes for LOGOFF routines.

When a task ends, all the LOGOFF routines are called starting with the top, or most recent, request on the stack. Each request is removed from the queue and freed.

Automation Task Command Processors

Commands written to run under OST tasks must consider the effects of running under automation tasks and MVS console tasks. These OSTs have the TVBAUTOO bit set to 1. This indicates that immediate commands and full screen mode commands are not supported in this task.

Installing a Command Processor

To install a command processor, define the command verbs with CMDMDL statements as described in the *NetView Installation and Administration Guide*. If your command processor will do line mode output (DSIPSS TYPE=OUTPUT), you should specify ECHO=Y. If your command processor does full screen mode output (DSIPSS TYPE=ASYPANEL), you should specify ECHO=N. Then assemble and link-edit the command processor into a load module in the NetView load library. NetView loads and calls the command processor according to its linkage editor attributes.

See "Preparing Your Code for Use" on page 4 for information on testing your command processor before use.

Template for a Command Processor

The following template illustrates basic entry and exit processing required by all command processors. It is available on-line in the NetView sample library (SYS1.CNMSAMP) under the name CNMS4202.

```
ATMPCMDP CSECT
***********************************************************************
*                                                                     *
* (C) COPYRIGHT IBM CORP. 1989                                        *
*                                                                     *
* IEBCOPY   SELECT MEMBER=((CNMS4202,ATMPCMDP,R))                     *
*                                                                     *
* MODULE NAME:                                                        *
*                                                                     *
* FUNCTION:                                                           *
*                                                                     *
*                                                                     *
* SYNTAX:                                                             *
*                                                                     *
*                                                                     *
* INSTALLATION:                                                       *
*                                                                     *
*                                                                     *
* INPUT:   REG 1 - ADDRESS OF COMMAND WORK BLOCK (DSICWB)             *
*          REG13 - ADDRESS OF CALLER'S SAVE AREA                      *
*          REG14 - RETURN ADDRESS                                     *
*          REG15 - ENTRY ADDRESS                                      *
*                                                                     *
* OUTPUT:                                                             *
*                                                                     *
*     REGISTERS:                                                      *
*          REG 0 - REG14 - RESTORED UPON RETURN                       *
*                                                                     *
*          REG 15 RETURN CODES:                                       *
*                 0  -  SUCCESSFUL                                    *
*                                                                     *
*                                                                     *
* NETVIEW MACROS:                                                     *
*                                                                     *
*          DSICBS - CONTROL BLOCK SERVICE                             *
*                                                                     *
***********************************************************************
```

Figure 10 (Part 1 of 4). Template for a Command Processor

```
         EJECT
         DSICBS DSICWB,DSIMVT,DSIPDB,DSISVL,DSISWB,DSITIB,DSITVB,      X
               PRINT=NO
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8                       MVT
R9       EQU   9                       TVB
R10      EQU   10                      TIB
R11      EQU   11                      CWB
R12      EQU   12                      BASE REG
R13      EQU   13                      SAVEAREA
R14      EQU   14
R15      EQU   15
         EJECT
***********************************************************************
*                                                                     *
*  SAVE REGISTERS AND ESTABLISH BASE REGISTER                         *
*                                                                     *
***********************************************************************
         USING *,R15
         B     PROLOG
         DC    C'ATMPCMDP  &SYSDATE. AT &SYSTIME.'
PROLOG   DS    0H
         STM   R14,R12,R12(R13)        SAVE REGISTERS
         DROP  R15
         LR    R12,R15                 SET BASE REGISTER
         USING ATMPCMDP,R12
```

Figure 10 (Part 2 of 4). Template for a Command Processor

```
************************************************************************
*                                                                      *
*  ESTABLISH ADDRESSABILITY TO THE COMMAND WORK BLOCK (CWB) AND SET    *
*  UP THE SAVE AREA USING CWBSAVEA                                     *
*                                                                      *
************************************************************************
         LR    R11,R1                  LOAD CWB ADDR
         USING DSICWB,R11              R11 BASE FOR COMMAND WORK BLOCK
         LA    R1,CWBSAVEA             USE CWBSAVEA FOR SAVEAREA
         ST    R1,8(R13)               STORE MY SA INTO CALLERS SA
         ST    R13,4(R1)               STORE CALLERS SA IN MINE
         LR    R13,R1                  R13 HAS MY SAVEAREA ADDRESS
************************************************************************
*                                                                      *
*  ESTABLISH ADDRESSABILITY TO THE TASK INFORMATION BLOCK (TIB), THE   *
*  TASK VECTOR BLOCK (TVB), AND THE MAIN VECTOR BLOCK (MVT).           *
*                                                                      *
************************************************************************
         L     R10,CWBTIB              GET DSITIB ADDRESS
         USING DSITIB,R10              ESTABLISH ADDRESSABILITY
         L     R9,TIBTVB               GET DSITVB ADDRESS
         USING DSITVB,R9               ESTABLISH ADDRESSABILITY
         L     R8,TVBMVT               GET DSIMVT ADDRESS
         USING DSIMVT,R8               ESTABLISH ADDRESSABILITY
*
         XC    CWBADATD,CWBADATD       ZERO AUTODATA AREA
         SLR   R15,R15                 ZERO RETURN CODE REGISTER
************************************************************************
*                                                                      *
*                                                                      *
*                                                                      *
*                                                                      *
*                                                                      *
*                                                                      *
*      MAIN          PROCESSING          GOES           HERE          *
*                                                                      *
*                                                                      *
*                                                                      *
*                                                                      *
*                                                                      *
*                                                                      *
************************************************************************
```

Figure 10 (Part 3 of 4). Template for a Command Processor

```
***********************************************************************
*                                                                     *
*  EXIT                                                               *
*                                                                     *
***********************************************************************
RETURN   EQU   *
         L     R13,4(R13)              GET CALLERS SAVEAREA ADDRESS
         L     R14,12(R13)             RESTORE REG14
         LM    R0,R12,20(R13)          RESTORE REGS
         BR    R14                     RETURN
         SPACE
         EJECT
         LTORG
         EJECT
*
DSICWB   DSECT
         ORG   CWBADATD                AUTODATA AREA
         DS    XL(256-(*-CWBADATD))    AUTODATA LENGTH CHECK
         END
```

Figure 10 (Part 4 of 4). Template for a Command Processor

Chapter 5

Chapter 5. Writing User Subtasks

Types of User Subtasks

NetView provides two methods for writing user subtasks. The first and recommended method uses the data services task (DST) interface as a coding base. The DST base provides interfaces for the following functions:

- An initialization user exit
- A subtask processing module (DSIZDST)
- CNMI service
- VSAM service
- Data Services Command Processor (DSCP)

The DST provides an ideal structure for user-written tasks since the DST can be defined with VSAM services, CNM services, or neither service (user-defined functions can be implemented within the data services command processors). The DST provides all the low-level user subtask functions that you would otherwise have to code if you wrote a complete optional substask. An optional subtask should only be written if access to the subtask ECB processing loop is required.

The other method requires that you code an optional (OPT) subtask. With this method, NetView supplies an intertask communication (message queue) ECB and a termination ECB. It is up to you to provide an appropriate ECB processing loop and any additional function. This requires more coding effort than the second method, but it allows for more flexibility as to what kinds of functions can be implemented.

Optional Subtask Processing

Overview

A user subtask requires the following processes:

- Installation
- Initialization
- Processing
- Termination

Installation The TASK statement is required to define an optional subtask to NetView. It is added to the DSIDMN memeber of DSIPARM. The DSIDMN member is processed during NetView initialization.

Initialization The initialization process of the subtask performs any required initialization functions. Examples would be acquiring NetView control blocks via the DSILCS macro and acquiring dynamic storage via the DSIGET macro.

Processing The processing part of a subtask should begin by invoking the DSIWAT macro to wait on an ECB list. The ECB list MUST include the subtask termination ECB (TVBTECB) and generally includes the message queue ECB (TVBMECB), used for intertask communication (via the DSIMQS macro). User defined ECBS can also be included in the ECB list.

Termination The termination process must free all acquired resources (storage, NetView control blocks, etc.) and return to NetView.

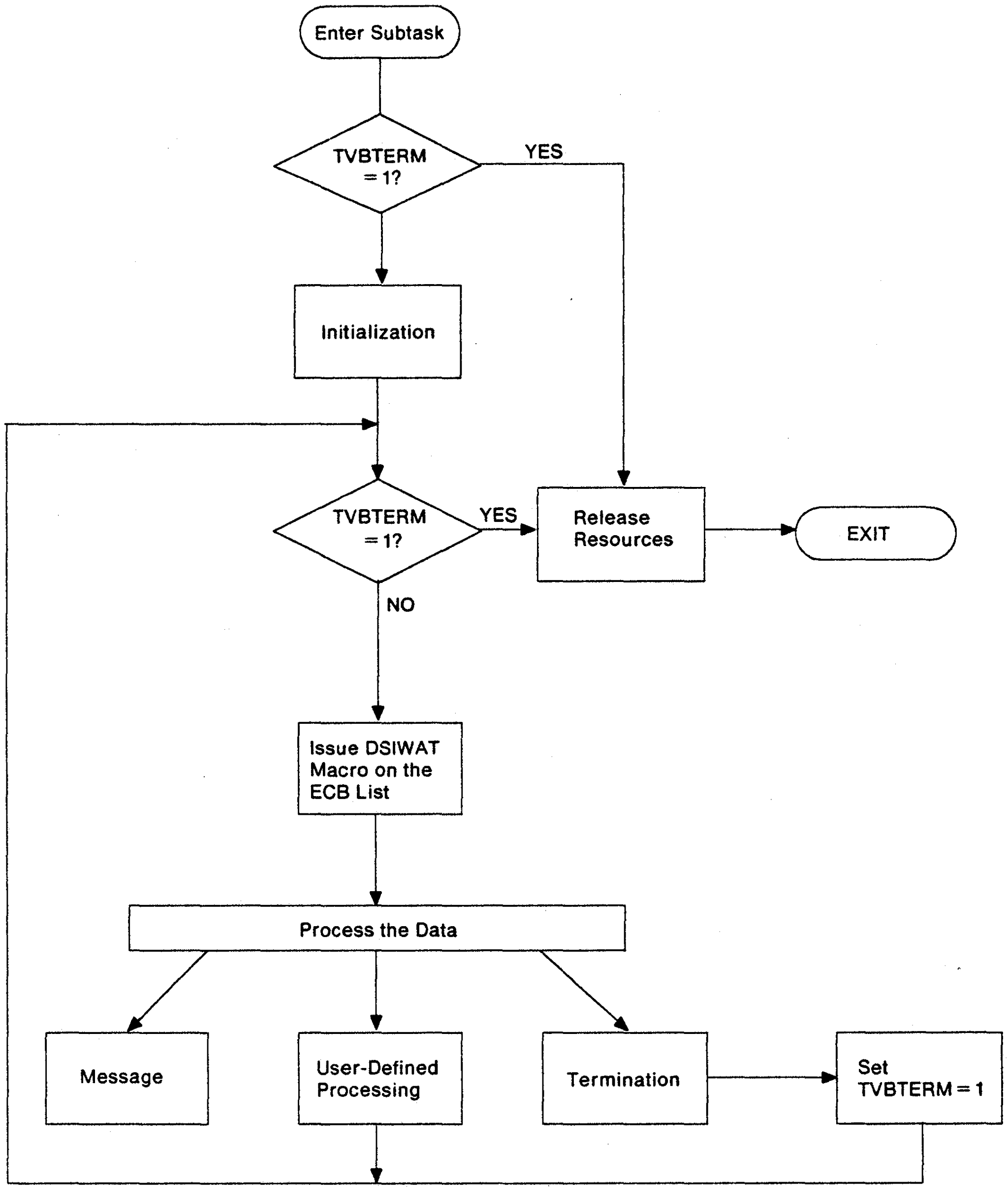


Figure 11. Subtask Organization

Installation

A TASK statement for the subtask must be coded in the DSIDMN member of the DSIPARM data set.

The TASK statement defines the task to NetView and provides the following information:

- MOD keyword - the name of the module to be run as a subtask. The module must be link-edited into the proper NetView library.
- TSKID keyword - The task name. Each task in NetView must have a unique task name.
- MEM keyword - Specifies the user-defined initialization member found in DSIPARM to be used by this task. The user is responsible for the format and contents of the specified member. The member can be read and processed during the task initialization.
- PRI keyword - Specifies the relative task priority (1-9). 1 is the highest task priority that can be assigned, and 9 is the lowest.
- INIT keyword - Specifies whether the task is to be started during NetView initialization (INIT=Y) or via the START command only (INIT=N).

```
TASK MOD=USERMOD,TSKID=USERTASK,MEM=USERMEM,PRI=7,INIT=Y
```

In the example above, the subtask identification is USERTASK. USERMEM is the name of the DSIPARM member (for MVS) or the file with file type NCCFLST (for VM) that contains additional initialization information. The priority of the subtask is 7, and the subtask will be started during NetView initialization. For more information on the TASK statement, see the *NetView Administration Reference*.

Initialization

Attaching the Subtask

Issuing the START TASK command or specifying INIT=Y on the TASK definition statement causes a normal attach. The TVBTERM bit in the TVB is set to off (0).

When an OPT subtask is attached, the registers contain the following information:

Register	**Contents**
1	The address of the task vector block (TVB)
13	The address of a standard 72-byte save area used to store the caller's registers
14	The return address
15	The entry address of the subtask
0, 2 - 12	Unspecified.

The control blocks used in attaching an OPT subtask are TVB, TIB, MVT, and SVL. The TVB contains the address of the TIB and the MVT, as well as the task control block of the operating system. The MVT contains the address of the SVL. Refer to Chapter 7 on page 119 for detailed descriptions of these control blocks.

After the subtask is initialized and before it starts processing, it must indicate that it is ready to begin processing by doing the following:

- Setting the TVBOPID field of the TVB to a unique subtask identifier
- Setting the TVBACTV bit to on.

One method of setting the TVBOPID field is to copy the contents of TVBLUNAM into TVBOPID. (TVBLUNAM is the value of the TSKID parameter in the TASK definition statement.) Another method is to use a predefined value.

TVBMEMNM Field

The value of the MEM keyword of the TASK definition statement is generally used as the name (one to eight characters) of an initialization member or file. The value of the MEM keyword is found in the TVBMEMNM field of the TVB if initialization input is required. If the subtask is coded to process this member or file, the following macros would be invoked to read from the member or file:

- DSIDKS TYPE=CONN,NAME=DSIPARM to connect the subtask to disk services for the DSIPARM dataset. (For VM, you would use NAME=NCCFLST.)
- DSIDKS TYPE=FIND,NAME=TVBMEMNM to find the member or file and read the first record.
- DSIDKS TYPE=READ to read a record. The READ would be repeated until a user-defined END statement is read or until an end-of-file return code is returned.
- DSIDKS TYPE=DISC to disconnect from disk services.

If the initialization member or filename is not used by the subtask, you can use TVBMEMNM for other purposes, depending on the manner in which the MEM keyword of the TASK statement is specified. For example, you may decide to use this field as the DD name to be opened by the subtask; or you can specify a default operator to receive messages.

Processing

ECB Loop

The processing section of a subtask usually begins by issuing the DSIWAT macro to wait on an event control block (ECB) list. The ECB list must contain the subtask termination ECB (TVBTECB), and generally contains the message queue ECB (TVBMECB), used for intertask communication via the DSIMQS (Message Queuing Service) macro. In addition to these two NetView provided ECBS, the ECB list can also contain user defined ECBS for additional functions.

Intertask Communication

If your task will receive messages or commands from other tasks (or exits), include the (normal) message ECB (TVBMECB) in your task's wait list. Optionally, you may wish to include also the high and low priority message ECBS (TVBMECBH and TVBMECBL). If the the task services the high and low queues, you should set the bit TVBMM to '1'B to indicate this. (When TVBMM is '0'B, the message queuing service will put all messages on the normal queue, regardless of how they were sent.) You should retest the high priority ECB after each item of work on a lower queue, to allow higher priority work to preempt the queue of lower priority work.

Each message queue should be thought of as a triplet: a private queue, a public queue, and an ECB. (There are two queues for reentrancy reasons.)

Table 3. Message Processing Triplets.

Priority	Private Queue	Public Queue	ECB
High	TVBMPRQH	TVBMPUBH	TVBMECBH
Normal	TVBMPRIQ	TVBMPUBQ	TVBMECB
Low	TVBMPRQL	TVBMPUBL	TVBMECBL

Message Queue Processing: When a message buffer is received by your subtask, the TVBMECB event control block will be posted and the message buffer will be inserted at the head of the public message queue pointed to by TVBMPUBQ (this is a LIFO queue). When your subtask detects that the TVBMECB ECB has been posted, the public message queue can be moved to the private message queue (TVBMPRIQ) for processing. Because the DSIMQS service handles situations such as main line interruption (i.e., an exit running asynchronously queues a message buffer to your subtask), simultaneous processing in multiple subtasks, and parallel processing in multiprocessor environments, the assembler compare and swap (CS) instruction must be used to acquire message buffers from the public message queue.

To process the public message queue, do the following:

1. Set TVBMECB to 0.
2. Use the assembler CS instruction to obtain the queue of buffers from TVBMPUBQ and store zero into TVBMPUBQ.
3. Reverse the order of the queue (make it FIFO) so that the message buffers can be processed in the order that they were actually received.

The following segment of assembler code demonstrates how to move the public message queue to the private message queue (addressability to the DSITVB control block is assumed):

```
MESSAGEQ EQU   *                  BRANCH HERE WHEN TVBMECB POSTED
         XC    TVBMECB,TVBMECB    CLEAR MESSAGE ECB
CHEKQ    EQU   *
         SLR   R0,R0              CLEAR SWAP REGISTER
         L     R3,TVBMPUBQ        LOAD COMPARAND REGISTER
         CS    R3,R0,TVBMPUBQ     CS ZERO ON THE PUBLIC QUEUE
         BNE   CHEKQ              RETRY IF TVBMPUBQ MODIFIED
         USING BUFHDR,R3          MAP BUFHDR ONTO QUEUE HEAD
REVQ     EQU   *                  MAKE LIFO QUEUE A FIFO QUEUE
         L     R1,HDRNEXTM        R1 = POINTER TO NEXT BUFFER
         ST    R0,HDRNEXTM        SET NEXT TO PREVIOUS
         LR    R0,R3              MAKE CURRENT PREVIOUS
         LTR   R3,R1              END OF QUEUE?
         BNZ   REVQ               CONTINUE UNTIL END REACHED
         ST    R0,TVBMPRIQ        ANCHOR THE PRIVATE QUEUE
PROCESS  EQU   *                  BEGIN BUFFER PROCESSING
```

The message buffers can be dequeued from the private message queue and processed. After each buffer is processed, it must be freed. Message buffers were obtained with DSIGET Q=NO and SUBPOOL 0 so they must be freed with DSIFRE Q=NO and SUBPOOL 0 (these are the default values).

Message Buffer Contents: Message buffers are discussed in detail in Chapter 2 on page 7. They may be actual messages to be displayed (HDRMTYPE = HDRTYPEU) or internal function requests (HDRMTYPE = HDRTYPEI). For internal function requests queued to your optional subtask, you can define your own function type by setting the IFRCODE field to IFRCODUS (user function) and then taking appropriate user-defined action when the buffer is received by your optional subtask.

Using IFRCODUS to Invoke a User-Defined Command Processor: The following technique should be used to call a user-defined command processor under your optional subtask:

1. Issue DSIMQS (from any subtask environment) to send an internal funtion request (HDRMTYPE=HDRTYPEI and IFRCODE=IFRCODUS) to your optional subtask. The command name and command parameters should follow the DSIIFR portion of the buffer passed to DSIMQS.
2. When processing the IFRCODUS message buffer under your subtask, add 2 to HDRTDISP to adjust the displacement to the start of the command and subtract 2 from HDRMLENG to keep the length consistent.
3. Follow the steps documented under "Calling a Command Directly" on page 21 to call the command processor.

The command processor called should only be defined as TYPE=D or TYPE=RD on its respective CMDMDL statement in DSICMD, and must be a user-written command processor. Do not call any NetView-provided command procedures from your optional subtask.

Note: The DSIZCSMS and DSIZVSMS macros cannot be used under your optional task.

Sending Message Buffers: Use the DSIMQS macro to send message buffers to other subtasks. These message buffers may contain actual operator messages to be displayed, or they may contain internal function requests (IFRS) to be executed by the receiving subtasks. See Chapter 8 on page 161 for details of the DSIMQS service.

Operator Communications

Sending Messages to Operators You may use DSIPSS TYPE=OUTPUT or TYPE=IMMED to send messages. Messages sent this way will go to the operator who started the task (owner). If the task was started during NetView initialization or the owner has logged off, the message is sent to the PPT for routing and automation.: The DSIMQS macro can also be used to send messages to the authorized receiver of messages or to the operator that started the subtask. The TIBMSGNM field of the DSITIB control block will contain zeroes if the task was started during NetView initialization (INIT=Y was specified on the TASK definition statement).

Note: Commands from operators are buffers with HDRMTYPE equal to HDRTYPET, HDRTYPEB, or HDRTYPQC. All other commands are HDRTYPEI.

Logging Messages: You can use macro DSIWLS to write messages from the subtask to the network log, the MVS system log, an external log, or a NetView sequential log. See "DSIWLS – Write Log Services" on page 214 for more information. Hard-copy logging may not be started for user-written subtasks.

Calling Command Processors See "Calling a Command Directly" on page 21.

Notes:

1. Command lists, immediate commands, and regular commands cannot be called from an optional task, only command processors defined as TYPE=D or TYPE=RD.
2. DST service macros DSIZVSMS and DSIZCSMS cannot be invoked under an optional task.

User-defined Functions: When a user-defined ECB is posted for work, a user-defined command processor can be called as explained in the previous section or a subroutine can be called to perform the requested function. It is up to you to decide how you want to handle your implemented function.

Termination

Terminating the OPT: When a subtask terminates normally, the TVBTERM bit is set to on, indicating that the subtask's resources should be released. When a subtask terminates abnormally, the TVBTERM bit is set to on and the subtask is reattached. (This is called a cleanup attach.) When the subtask regains control, it frees the resources it had obtained and exits normally.

Include the TVBTECB field of TVB in the subtask ECB list for each OPT subtask you write. When a CLOSE NORMAL command is issued and all operators have logged off, the main task posts the TVBTECB of the subtask. This posting indicates that subtask termination is requested. When the subtask finds the TVBTECB posted, the subtask must perform the following:

- Release all resources (See Releasing Queued Storage below.)
- Set the TVBOPID field to blanks
- Set the TVBACTV bit to off
- Set the TVBTERM bit to on
- Reload the registers originally passed and return to NetView.

After releasing all resources, no NetView macros may be issued.

Releasing Queued Storage: The DSIGET Q=YES option enables storage to be easily freed for both normal and abnormal subtask termination. During a subtask termination, use DSIFRE AQ=YES to free any remaining storage obtained by DSIGET Q=YES.

To release all queued storage, issue DSIFRE AQ=YES. Be certain that any VTAM ACBS owned by this task have been closed before issuing DSIFRE AQ=YES. NetView will free all queued storage with one invocation of DSIFRE AQ=YES (both mainline and exit storage). Some macros may require queued storage; therefore, the subtask may not issue any NetView macros after releasing the queued storage.

Additional Considerations

Special Requirements for IRB Exits: User-written IRB exits that invoke NetView macro services require the following special processing:

- On entry, if the TVBINXIT is not on, then set it on. If the TVBINXIT bit is already set to on, increment TIBMUXIT by 1.
- On exit, if TIBMUXIT is zero, then clear the TVBINXIT bit. However, if TIBMUXIT is greater than zero, then you must decrement it by 1.

Displaying Status: The LIST command displays the status of a subtask on an operator's terminal. For OPT subtasks, in addition to status, a header line and the contents of TVBOPID and TVBLUNAM are also displayed. Status is determined by the following TVB bit fields in the following order:

1. TVBRCVRY — recovering
2. TVBLGOFF — stopping
3. TVBACTV — active
4. TVBLGON — starting
5. None of the above — inactive.

The subtask can also create its own status display.

VTAM Outage Processing: User-written code (exit routines, command processors, and subtasks) requiring VTAM will get error codes whenever VTAM is inactive. User-written subtasks that require VTAM must provide for the case of VTAM ending without NetView ending in these ways.

- If your task opens a VTAM ACB, you can code a TPEND exit for VTAM that is called when VTAM ends. Your TPEND exit can post TVBTECB to signal task termination to begin.
- If your task requires VTAM to be active and does not open an ACB, you can still be notified. Set the TVBAUTVE bit in the TVB for your task. When NetView's main task TPEND is called (for HALT NET, QUICK, for HALT NET, CANCEL, and for VTAM ABEND), the NetView main task will post TVBTECB for every task that has TVBAUTVE set to 1.
- In either case, NetView also provides another bit, TVBAUTVS, which causes NetView main task to reattach your subtask when NetView detects that VTAM has been reactivated to the point that the main task's ACB was opened successfully. Set TVBAUTVS to 1 for this function.

Data Services Task (DST)

Overview

A data services task is a set of NetView interfaces built on top of the optional task base. NetView provides a subtask processing module (DSIZDST) along with the following:

- An initialization exit interface
- A Data Services Command Processor (DSCP) interface that provides the following services:
  - A CNM data services macro interface (DSIZCSMS) to request and send data across the Communication Network Management interface.
  - An interface to allow a command processor to receive unsolicted CNM data
  - A VSAM data services macro interface (DSIZVSMS) to PUT and GET records from a pre-defined VSAM dataset.
- Various user exit interfaces

Installation

A TASK statement for the subtask must be coded in the DSIDMN member of the DSIPARM data set. This TASK statement follows the same format as the optional task TASK statement with the following exceptions:

1. The MOD keyword must specify DSIZDST as the subtask processing module. DSIZDST is provided by NetView and provides the necessary initialization, processing, and termination routines to use the DSCP interfaces.
2. The initialization dataset member (specified by the MEM keyword) must contain DSTINIT statements to provide various initialization parameters required by DSIZDST. The statements will be discussed below under their respective interfaces.

The values of the other TASK statement keywords have the same meaning as those coded for an optional task.

- PRI keyword - Specifies the relative task priority (1-9). 1 is the highest task priority that can be assigned, and 9 is the lowest.
- INIT keyword - Specifies whether the task is to be started during NetView initialization (INIT=Y) or via the START command only (INIT=N).
- TSKID keyword - The task name. Each task in NetView must have a unique task name.

```
TASK MOD=DSIZDST,TSKID=USERTASK,MEM=USERMEM,PRI=7,INIT=Y
```

In the example above, the subtask identification is USERTASK. USERMEM is the name of the DSIPARM member (for MVS) or the file with file type NCCFLST (for VM) that contains additional initialization information. The Priority of the subtask is 7, and the subtask will be started during NetView initialization.

Initialization

For additional details on DSTINIT statements, see *NetView Administration Reference*.

DSTINIT Keywords: The following keywords apply to DST initialization processing.

- FUNCT - The FUNCT keyword specifies which DST services will be required. In all cases, the ability to call DSCPs is provided. The function choices are:
  - OTHER - The DST does not require the CNMI or VSAM interfaces.
  - BOTH - Both the VSAM and CNMI interfaces are required.
  - CNMI - Only the CNMI interface is required.
  - VSAM - Only the VSAM interface is required.
- XITDI - The XITDI keyword specifies the name of the user provided initialization exit. The exit is called with the standard NetView user exit interface as documented in Chapter 3 on page 31 and is called once for every statement in the specified initialization member (MEM keyword of TASK statement). When End-Of-File has been reached, USERPDB and USERMSG will both be 0. For each statement (except End-Of-File condition), the standard user exit return codes will cause the following actions:

- USERASIS (0) - The statement will be processed by the NetView DST module (DSIZDST). If it is not a valid DSTINIT statement, DSIZDST will reject it with an error message and continue processing.
- USERDROP (4) - The statement will not be processed by DSIZDST. This return code should be used if your user exit is going to process the statement (you can define your own initialization statements).
- USERSWAP (8) - The swapped buffer will be processed by DSIZDST. If the swapped buffer does not contain a valid DSTINIT statement, it will be rejected by DSIZDST and processing will continue.

When returning from the last call (for End-Of-File), any non-zero return code will terminate the DST. This should only be done if the initialization process has failed.

The initialization exit should invoke the DSIPUSH service to define a LOGOFF routine. The LOGOFF routine will be invoked during normal or abnormal end of task processing (no termination exit is provided). The LOGOFF routine should free any resources that the user has acquired. Storage that has been acquired with the Q=YES option is automatically freed by the DSIZDST module.

Data Services Command Processor

A data services command processor (DSCP) generally performs CNM data services, using macro DSIZCSMS, or VSAM data services, using macro DSIZVSMS, or both services.

When a DST calls a DSCP, the input to the DSCP includes the address of a data services request block (DSRB) in the CWBDSRB field. The DSRB function code (DSRBFNCD) indicates the purpose for which the command was called. DSRBFNCD is described under "DSRB — Data Services Request Block" on page 129.

There are two restrictions to observe when writing a DSCP:

- Only commands defined as TYPE=D or TYPE=RD may be called under a DST or queued to a DST. Call only user-defined commands directly. Commands called from your DSCP cannot use NetView macros DSIZVSMS nor DSIZCSMS.
- Use only DSIPSS TYPE=OUTPUT or TYPE=IMMED. Messages sent this way will go to the operator who started the task (owner). If the task was started during NetView initialization or if the owner has logged off the message is sent to the PPT for routing and automation.

Data services requests are generally sent with HDRMTYPE=HDRTYPEI. Operators can queue commands using the EXCMD command, and these commands may be identified since they will be HDRTYPET, HDRTYPEB, or HDRTYPQC. You may wish to check the HDRMTYPE field and reject direct operator requests.

CNM Data Services

The DST provides access to both solicited and unsolicited CNM data, DSIZCSMS can be issued by a DSCP to solicit CNM data from the network. A DSCP can be defined to receive unsolicited data from VTAM.

An ACB with AUTH=CNM must be defined to VTAM with the ACB name matching the task ID of the DST.

Unsolicited CNM Data Interface

VTAM provides a default table (ISTMGC01) which controls the routing of unsolicited CNM RUs. You can write a supplemental table (ISTMGC00) to override the default routing information provided by VTAM. The routing information consists of a particular RU type and the name of an application which is to receive the particular type of data. When a DST is defined with CNMI services, an ACB is opened with an ACB name (the application name) equivalent to the task name as defined by the TSKID parameter of the DST TASK definition statement (the one exception is Hardware Monitor whose CNMI DST's task name is BNJDSERV, but the application name is BNJHWMON). If the DST task name is entered as the application name in the VTAM routing table, the unsolicited data RU will be passed to the unsolicited data services command processor for that DST.

DSTINIT Keywords

- UNSOL - Specifies the command verb name of the module that is to serve as the unsolicited DSCP for this DST. The unsolicited DSCP should not issue the DSIZCSMS macro, but may issue the DSIZVSMS macro.
- DSRBU - Specifies the number of unsolicited DSRBs which are to be allocated to this DST. If unsolicited CNM data isn't going to be processed by this DST, then this value should be set to zero. If the unsolicited DSCP is going to issue the DSIZVSMS macro, then this value should be set to the number of concurrent DSIZVSMS requests which are to be allowed. If the unsolicited DSCP is not going to issue the DSIZVSMS macro, then this value should be set to 1.

Note: To issue DSIZVSMS also, FUNCT=BOTH must be specified.

DSCP Interface: When the unsolicited DSCP receives control, the DSRBFNCD field will contain the DSRBFUNS (unsolicited) function code, DSRBUBUF will be zero, and DSRBCUSB will contain the address of a NetView buffer containing the unsolicited data. The RU starts at the offset specified in HDRTDISP and the RU length is in HDRMLENG. If a Deliver header is present, it will be considered part of the data (i.e. - HDRTDISP will point to the start of the Deliver header). See the *VTAM Programming* book for more information.

The return codes on entry to the unsolicited DSCP are as follows:

DSRBRCMA	DSRBRCMI	Meaning
00	00	Successful completion.
00	16	User exit rejected the Deliver RU. HDRMLENG is set to zero.
00	20	Data has been truncated. The length of the Deliver RU was greater than the length of the buffer. HDRMLENG is set to the truncated length.
00	24	Data was truncated after the user exit returned with a return code of USERSWAP. HDRMLENG is set to the truncated length.

Solicited CNM Data Interface

The DSIZCSMS macro can be invoked by a DSCP to acquire Communications Network Management data from the network.

DSTINIT Keyword: DSRBO - Specifies the number of solicited DSRBs that will be required by this task and limits the number of concurrent DSIZCSMS and/or DSIZVSMS requests. This value must be at least 1 (a DSCP will not be called unless a solicited DSRB is available) and no greater than 862.

DSCP Interface: Acquiring CNM data is a two part process. When the DSCP is first driven (generally by a command buffer MQSed by an OST task), the DSRBFNCD field will contain a value of DSRBFNRM. The CWBDSRB field will point to a DSRB that must be passed on the DSIZCSMS macro. The first step is to issue the DSIZCSMS macro with the supplied DSRB. After the macro is issued, register 15 will contain the major return code and register 0 will contain the minor return code (additional completion information). If register 15 is not zero then the macro has failed. If register 15 is 0 then the request has successfully been sent to VTAM. At this time, the DSCP should be exited because the data will be returned on a subsequent invocation of the same DSCP (this is called a re-drive operation).

When the DSCP is re-driven (the second part of the process), the DSRBFNCD code will be DSRBFSOL. The DSRBRCMA (DSRB Major Return Code) and the DSRBRCMI (DSRB Minor Return Code) must be checked to see if the request completed successfully.

DSRBRCMA	DSRBRCMI	Meaning
00	00	Successful completion.
00	04	Negative response was received. DSRBINPT contains the address of the negative response.
00	08	Insufficient storage to process the request.
00	16	User exit rejected the Deliver RU. HDRMLENG is set to zero.
00	20	Data has been truncated. The length of the Deliver RU was greater than the length of the buffer. HDRMLENG is set to the truncated length.
00	24	Data was truncated after the user exit returned with a return code of USERSWAP. HDRMLENG is set to the truncated length.
00	28	VTAM rejected the request.
00	32	CNM interface closed due to unrecoverable error.
00	36	Positive response was received.
00	44	Cancellation due to timer completion. This code is returned only when running with VTAM V3R1.1 or later.

If the request completes successfully, the input buffer supplied on the initial DSIZCSMS invocation (INPUT parameter) will contain the received data. The buffer will contain a standard NetView buffer header with HDRTDISP containing the offset to the start of the data. If the data is preceded by a Deliver RU, HDRTDISP will contain the offset to the start of the Deliver RU.

After the initial invocation of DSIZCSMS and until the DSCP is redriven, the DSRB is considered 'in use' and is not available to other DSCPs (other DSCPs can run during this time frame only if the DSRBO value is greater than 1 and there is a DSRB that is 'not in use' by another DSCP). When the DSCP is re-driven, the DSRB is the only control block that is the same as on the initial invocation of the DSCP. The DSRBUSER field has been provided for your use and can be used to contain or point to any additional environment information that you wish to maintain.

VSAM Service Interface

The DSIZVSMS macro can be invoked by a DSCP to perform I/O to a specified VSAM data set.

DSTINIT Keywords: The primary and secondary data sets are user-defined and are switchable.

- PDDNM - Specifies the DD name of the primary data set to be used by VSAM services. This data set must be allocated prior to starting the DST.

- PPASS - Specifies the VSAM password to be used when the primary data set ACB is opened.
- SDDNM - Specifies the DD name of the secondary data set to be used by VSAM services. This data set must be allocated prior to starting the DST. The NetView SWITCH command is used to control which data set is currently the 'active' data set.
- SPASS - Specifies the VSAM password to be used when the secondary data set ACB is opened.
- MACRF - Specifies local resource sharing.
- XITVN - Specifies a user exit to receive control when an empty VSAM data set has been opened for processing. This exit allows you to put an initialization record into the data set.
- XITVI - Specifies a user exit to receive control upon input from the VSAM data set before the input record is passed to the requesting DSCP.
- XITVO - Specifies a user exit to receive control before output of a record to the VSAM data set.

Note: If DSRBO is greater than 1, then NetView does not guarantee that the DSIZVSMS requests for VSAM Puts will be processed in the order that they were submitted. (The requests will complete asynchronously.)

DSCP Interface: Like the CNM Interface service (DSIZCSMS), using DSIZVSMS is a two part process. When the DSCP is first driven (generally by a command buffer MQSed by an OST task), the DSRBFNCD field will contain a value of DSRBFNRM. The CWBDSRB field will point to a DSRB that must be passed on the DSIZVSMS macro. The first step is to issue the DSIZVSMS macro with the supplied DSRB. After the macro is issued, register 15 will contain the major return code and register 0 will contain the minor return code (additional completion information). If register 15 is not zero then the macro has failed. If register 15 is 0 then the request has successfully been sent to VSAM. At this time, the DSCP should be exited because the success or failure of the VSAM I/O requested will be returned on a subsequent invocation of the same DSCP (this is called a re-drive operation). When the DSCP is re-driven (the second part of the process), the DSRBFNCD code will be DSRBFVSM. The DSRBRCMA (DSRB Major Return Code) and the DSRBRCMI (DSRB Minor Return Code) must be checked to see if the request completed successfully.

The return codes are as follows:

DSRBRCMA	DSRBRCMI	Meaning
00	00	Successful completion.
00	16	User exit processing of VSAM input has rejected the input. HDRMLENG is set to zero.
00	24	Data has been truncated. User exit returned data longer than NetView buffer on RC = USERSWAP. HDRMLENG is set to the truncated length.
00	28	Invalid return code from user exit.
08		VSAM RPL feedback VSAM logical error, indicated in DSRBRCMI. See *OS/VS VSAM Programmer's Guide*.
12		VSAM RPL feedback VSAM physical error, indicated in DSRBRCMI. See *OS/VS VSAM Programmer's Guide*.

Relevant DSRB fields are as follows:

DSRBVRPL	The address of the VSAM RPL that was used for the I/O.
DSRBVACB	The address of the VSAM ACB for the DST.
DSRBVDAD	The address of the VSAM I/O buffer, with a standard BUFHDR. For GET requests, the BUFHDR HDRMLENG field indicates the length of the data read. HDRTDISP contains the offset to the data.
DSRBVKEY	The address of the key in the DSRBVDAD buffer.
DSRBVKLN	The key length.
DSRBVRTP	Indicates the type of request just completed: 1 - DSRVGET (VSAM GET) 2 - DSRVPUT (VSAM PUT) 3 - DSRVPNT (VSAM POINT) 4 - DSRVERS (VSAM ERASE) 5 - DSRVNRQ (VSAM ENDREQ).

After the initial invocation of DSIZVSMS and until the DSCP is redriven, the DSRB is considered 'in use' and is not available to other DSCPs (other DSCPs can run during this time frame only if the DSRBO value is greater than 1 and there is a DSRB that is 'not in use' by another DSCP). When the DSCP is re-driven, the DSRB is the only control block that is the same as on the initial invocation of the DSCP. The DSRBUSER field has been provided for your use and can be used to contain or point to any additional environment information that you wish to maintain.

Example of DSCP Design

A DSCP can be used to solicit communication network management (CNM) data from a resource in the network and record the results on a VSAM data set. To accomplish this, DSCP processing can follow the steps that are listed below and referenced in Figure 12 on page 93.

1. A DST subtask that receives an IFRCODCR buffer calls the DSCP with a function code of "initial call." The DSCP issues DSIZCSMS to solicit the CNM data. The DSCP returns to the caller. **1**
2. The DSCP is redriven[3] with a function code of "solicited CNM data." The DSCP issues DSIZVSMS to write the data to the VSAM data set. The DSCP returns to the caller. **2**
3. The DSCP is redriven with a function code of "VSAM I/O completed." If more than one CNM buffer is needed, the DSCP issues another DSIZCSMS to retrieve the buffer. The DSCP returns to the caller. **3**
4. Steps 2 and 3 repeat alternately until the DSCP has retrieved all of the data. **4**
5. When the DSCP is redriven for completion of the last VSAM PUT, it sends a completion message and returns control to the DST. Since neither DSIZCSMS nor DSIZVSMS was issued, the DSRB will not be redriven. Processing for the DSCP ends. **5**

[3] When a DSCP is redriven, its input control blocks and fields, except for the DSRB, may be completely different than those used in its previous invocation.

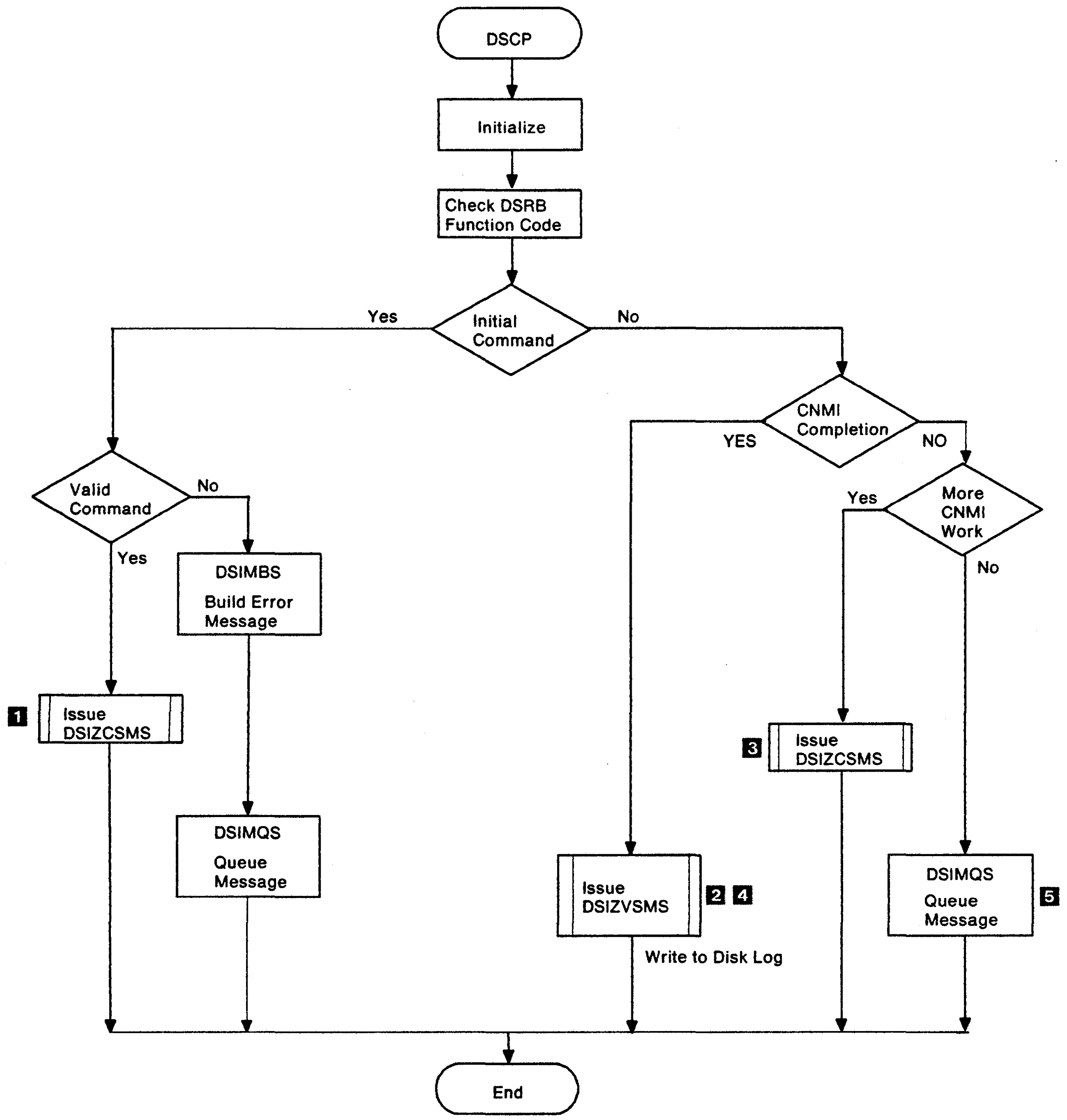


Figure 12. Structure of a Data Services Command Processor

Figure 13 shows one way you can structure data services requests. This example starts with an initial operator command. This invokes the DSCP using DSIMQS with an IFRCODCR buffer. When the DSCP has obtained VSAM or CNM data to be presented, it may do either of the following:

- Send the message data to the terminal (for standard or title-line output)
- Invoke a presentation services command processor (PSCP) to present the data (for full-screen output).

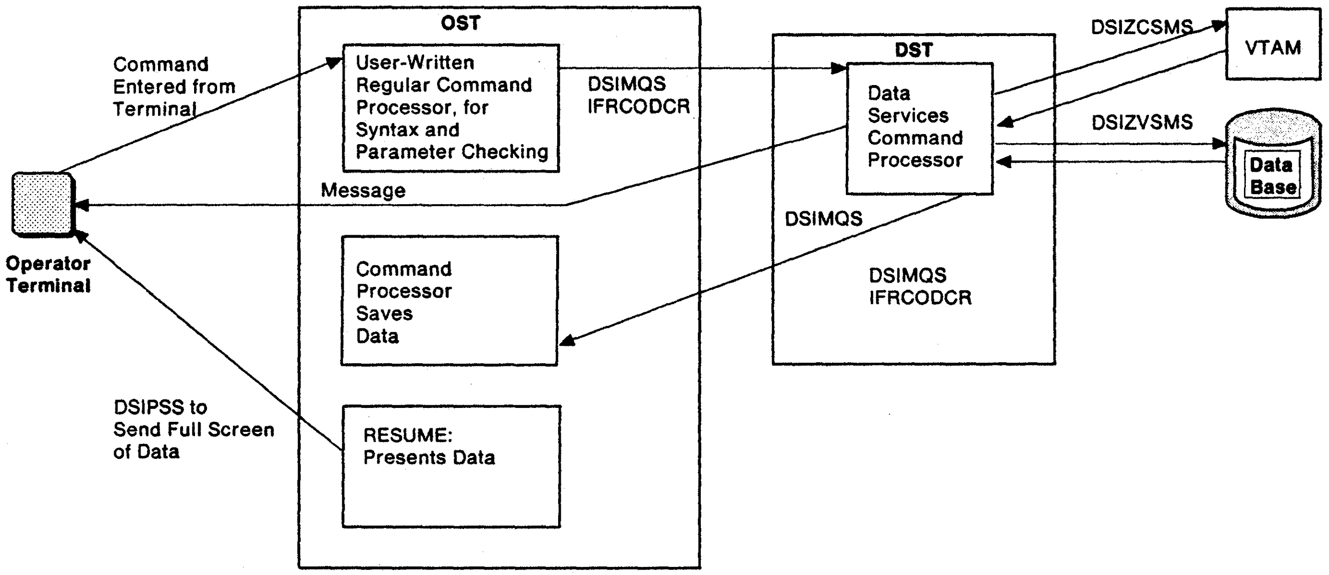

OST
Command Entered from Terminal
User-Written Regular Command Processor, for Syntax and Parameter Checking
DSIMQS IFRCODCR
Message
Operator Terminal
Command Processor Saves Data
DSIPSS to Send Full Screen of Data
RESUME: Presents Data
DST
Data Services Command Processor
DSIMQS
DSIMQS IFRCODCR
DSIZCSMS
VTAM
DSIZVSMS
Data Base

Figure 13. Example of Program Design for Data Services Requests

The command uses DSIGET to obtain storage. The address of this storage can be saved in DSRBUSER, or DSIPUSH can save it as a named storage pointer. This establishes and maintains data from one DSCP call to the next.

If the DSCP does not use the parse buffer PDB, you can improve the performance by specifying PARSE=N on the CMDMDL statement defining the DSCP. In this case, the command buffer is not parsed and no PDB is provided to the command processor. (PARSE=N applies only to commands invoked by a DST.)

An operator may have one or more pending DST requests. You can use the LIST DST command to list active DST requests.

User Defined Services

Command processors defined as TYPE=D or TYPE=RD can be invoked under the DST to perform user functions. They will be invoked with a standard NetView command processor interface (Register 1 will point to a DSICWB control block). If no parsing of the command buffer is required, then PARSE=N should be specified on the respective CMDMDL statement for the DSCP (this will improve performance).

Termination

You should issue DSIPUSH to set up a LOGOFF routine in your initialization exit. If the DST is terminated (normally or abnormally), the LOGOFF routine will be invoked and you should clean up any storage or resources you have acquired. Queued storage will be automatically released by the DSIZDST module prior to termination.

Template for an Optional Task

The following template illustrates basic initialization, ECB loop, and termination processing required by optional tasks. This template is available on-line in the NetView sample library (SYS1.CNMSAMP) as CNMS4277.

```
AOPTTSK CSECT
***********************************************************************
***********************************************************************
***                                                                 ***
***  IEBCOPY   SELECT MEMBER=((CNMS4277,AOPTTSK,R))                 ***
***                                                                 ***
***  (C) COPYRIGHT IBM CORP. 1989                                   ***
***                                                                 ***
***                                                                 ***
***  MODULE NAME: AOPTTSK                                           ***
***                                                                 ***
***  FUNCTION: USER DEFINED OPTIONAL SUBTASK.  THIS SUBTASK WAITS   ***
***  ON TASK TERMINATION AND THE MESSAGE QUEUE ECBS.  MESSAGES      ***
***  CAN BE PLACED ON THIS TASKS MESSAGE QUEUE BY:           .      ***
***                                                                 ***
***         1. USE EXCMD COMMAND                                    ***
***         2. USER WRITTEN COMMAND PROCESSOR WHICH SENDS BUFFER    ***
***            VIA DSIMQS MACRO WITH A BUFFER TYPE OF 'T'.          ***
***                                                                 ***
***********************************************************************
***********************************************************************
***                                                                 ***
***  INSTALLATION:                                                  ***
***                                                                 ***
***     1. ASSEMBLE AND LINKEDIT INTO NETVIEW LOAD LIBRARY          ***
***                                                                 ***
***     2. ADD TASK STATEMENT TO DSIDMN MEMBER IN DSIPARM           ***
***                                                                 ***
***        TASK MOD=AOPTTSK,TSKID=MYTASK,PRI=8,INIT=N               ***
***                                                                 ***
***  NOTE: TASK CAN BE STARTED AND STOPPED BY ISSUING               ***
***        START TASK=MYTASK OR STOP TASK=MYTASK.                   ***
***                                                                 ***
***                                                                 ***
***********************************************************************
* INPUT:   REG 1 - ADDRESS OF TASK VECTOR BLOCK (DSITVB)              *
*          REG13 - ADDRESS OF CALLER'S SAVE AREA                      *
*          REG14 - RETURN ADDRESS                                     *
*          REG15 - ENTRY ADDRESS                                      *
*                                                                     *
* OUTPUT:                                                             *
*                                                                     *
*     REGISTERS:                                                      *
*          REG 0 - REG14 - RESTORED UPON RETURN                       *
*                                                                     *
*          REG 15 RETURN CODES:                                       *
*                  0  -  SUCCESSFUL                                   *
*                                                                     *
*                                                                     *
* NETVIEW MACROS:                                                     *
*                                                                     *
*          DSICBS   - CONTROL BLOCK SERVICE                           *
*          DSIDATIM - DATE AND TIME SERVICE                           *
*          DSIFRE   - FREEMAIN STORAGE SERVICE                        *
*          DSIGET   - GETMAIN STORAGE SERVICE                         *
*          DSILCS   - LOCATE CONTROL BLOCKS                           *
*          DSIWAT   - ECB WAIT SERVICE                                *
***********************************************************************
```

Figure 14 (Part 1 of 11). Template for an Optional Task

```
************************************************************************
         EJECT
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6                        WORKAREA BASE REGISTER
R7       EQU   7                        DSIMVT BASE REGISTER
R8       EQU   8                        DSITVB BASE REGISTER
R9       EQU   9                        DSITIB BASE REGISTER
R10      EQU   10
R11      EQU   11
R12      EQU   12                       BASE REGISTER
R13      EQU   13
R14      EQU   14
R15      EQU   15
         EJECT
         PRINT OFF
         DSICBS DSIMVT,DSISVL,DSISWB,DSITIB,DSITVB
         PRINT ON
         USING *,R15
         B     PROLOG
         DC    C'AOPTTSK   &SYSDATE. AT &SYSTIME.'
PROLOG   DS    0H
         STM   R14,R12,12(R13)          SAVE REGS IN CALLERS SAVEAREA
         DROP  R15
         LR    R12,R15                  GET EPA FOR BASE REG
         USING AOPTTSK,R12              R12 IS BASE FOR PGM
************************************************************************
*                                                                      *
* ESTABLISH CONTROL BLOCK ADDRESSABILITY                               *
*                                                                      *
************************************************************************
         LR    R8,R1                    GET TVB ADDRESS
         USING DSITVB,R8                SET UP BASE FOR TVB
         L     R9,TVBTIB                GET TIB ADDRESS
         USING DSITIB,R9                SET UP BASE FOR TIB
         L     R7,TVBMVT                GET MVT ADDRESS
         USING DSIMVT,R7                SET UP BASE FOR MVT
************************************************************************
*                                                                      *
* ESTABLISH SAVE AREA                                                  *
*                                                                      *
************************************************************************
         ST    R13,TIBSAVES+4           SAVE CALLERS R13 IN MY SAVEAREA
         LA    R2,TIBSAVES              POINT R2 TO MY SAVEAREA
         ST    R2,8(0,R13)              PUT MY SAVEAREA ADDR IN CALLERS
         LR    R13,R2                   SET UP R13 TO MY SAVEAREA
************************************************************************
*                                                                      *
* CHECK FOR TERMINATION REDRIVE.                                       *
*                                                                      *
************************************************************************
         TM    TVBIND1,TVBTERM          WAS I DRIVEN FOR TERMINATION?
         BO    TERMINAT                 YES, GO AND DO TERM PROCESSING
```

Figure 14 (Part 2 of 11). Template for an Optional Task

```
***********************************************************************
*                                                                     *
* CALL INITRTN TO PERFORM INITIALIZATION PROCESSING.  TVBTERM WILL    *
* BE SET IF ANY ERRORS OCCURED DURING INITIALIZATION.                 *
*                                                                     *
***********************************************************************
         BAL   R14,INITRTN              TELL NETVIEW THAT I'M ACTIVE
         TM    TVBIND1,TVBTERM          ANY ERRORS?
         BO    RETURN                   YES, EXIT
***********************************************************************
*                                                                     *
* SET UP ECB LIST.  TERMINATION ECB (TVBTECB) AND THE MESSAGE QUEUE   *
* ECB (TVBMECB) ARE INCLUDED.                                         *
*                                                                     *
***********************************************************************
         LA    R1,TVBTECB               GET ADDRESS OF TERMINATION ECB
         ST    R1,ECBLIST               STORE IT IN ECB LIST
         LA    R1,TVBMECB               GET ADDRESS OF MESSAGE QUEUE ECB
         ST    R1,ECBLIST+4             STORE IT IN ECB LIST
         OI    ECBLIST+4,X'80'          MARK END OF LIST
***********************************************************************
*                                                                     *
* WAIT ON ECB LIST FOR TERMINATION OR MESSAGE EVENT.                  *
*                                                                     *
***********************************************************************
WAIT     EQU   *
         DSIWAT ECBLIST=ECBLIST         WAIT ON TERMINATION ECB
***********************************************************************
*                                                                     *
* CHECK FOR TERMINATION.                                              *
*                                                                     *
***********************************************************************
         TM    TVBTECB,TIBECBPO         TASK TERMINATION ECB POSTED
         BZ    MESSQ                    IF NOT, DO THE MESSAGE QUEUE
         XC    TVBTECB,TVBTECB          CLEAR TERMINATION ECB
         B     TERMINAT                 EXIT
***********************************************************************
*                                                                     *
* IF TERMINATION ECB NOT POSTED, ASSUME MESSAGE ECB WAS POSTED.       *
*                                                                     *
***********************************************************************
MESSQ    EQU   *
         XC    TVBMECB,TVBMECB          CLEAR MESSAGE ECB
***********************************************************************
*                                                                     *
* COMPARE AND SWAP THE PUBLIC QUEUE (TVBMPUBQ) TO THE PRIVATE QUEUE   *
* (TVBMPRIQ) FOR PROCESSING.                                          *
*                                                                     *
***********************************************************************
CHEKQ    EQU   *
         SLR   R0,R0                    ZERO THE SWAP REGISTER
         L     R3,TVBMPUBQ              LOAD COMPARAND REGISTER
         CS    R3,R0,TVBMPUBQ           CS ZERO ON THE QUEUE
         BNE   CHEKQ                    RETRY IF TVBMPUBQ CHANGED
```

Figure 14 (Part 3 of 11). Template for an Optional Task

```
***********************************************************************
*                                                                     *
* REVERSE THE PRIVATE QUEUE ORDER SO THAT THE MESSAGES ARE PROCESSED  *
* IN THE CORRECT ORDER.                                               *
*                                                                     *
***********************************************************************
         USING BUFHDR,R3
REVQ     EQU   *
         L     R1,HDRNEXTM              REVERSE
         ST    R0,HDRNEXTM               THE
         LR    R0,R3                       QUEUE
         LTR   R3,R1                        ORDER.  TEST FOR END OF Q
         BNZ   REVQ                     CONTINUE IF NOT END OF Q
         ST    R0,TVBMPRIQ              POINT PRIVATE ANCHOR AT FIFO Q
         DROP  R3
***********************************************************************
*                                                                     *
* CALL PROCMSG TO PROCESS THE PRIVATE QUEUE.                          *
*                                                                     *
***********************************************************************
         BAL   R14,PROCMSG              GO PROCESS THE MQS MESSAGE
         B     WAIT                     GO BACK AND WAIT
***********************************************************************
*                                                                     *
* BEGIN TERMINATION PROCESSING                                        *
*                                                                     *
***********************************************************************
TERMINAT EQU   *
         OC    TVBUFLD,TVBUFLD          ANY STORAGE TO FREE?
         BZ    RETURN                   NOPE, EXIT
         BAL   R14,TERMRTN              DO TERMINATION PROCESSING
         DSIFRE LV=4096,A=TVBUFLD       FREE STORAGE
         LTR   R15,R15                  DID I FREE THE STORAGE?
         BZ    FREEQ                    YES, FREE QUEUED STORAGE
         MVC   MESSAGE(L'FRUFAIL),FRUFAIL
         MVC   MESSAGE+7(8),TVBOPID   GET TASK ID
         LA    R1,L'FRUFAIL           GET MESSAGE ADDRESS
         BAL   R14,SENDMSG            DISPLAY MESSAGE
***********************************************************************
*                                                                     *
* FREE ALL QUEUED STORAGE.                                            *
*                                                                     *
***********************************************************************
FREEQ    EQU   *
         DSIFRE AQ=YES                  FREE QUEUED STORAGE
         LTR   R15,R15                  DID IT WORK?
         BZ    RETURN                   YES, QUEUED STORAGE WAS FREED
         MVC   MESSAGE(L'FRQFAIL),FRQFAIL
         MVC   MESSAGE+7(8),TVBOPID   GET TASK ID
         LA    R1,L'FRQFAIL           GET MESSAGE ADDRESS
         BAL   R14,SENDMSG            DISPLAY MESSAGE
```

Figure 14 (Part 4 of 11). Template for an Optional Task

```
***********************************************************************
*                                                                     *
* RETURN TO CALLER.                                                   *
*                                                                     *
***********************************************************************
RETURN   EQU   *
         L     R13,TIBSAVES+4          RESTORE CALLERS SAVEAREA ADDR
         SLR   R15,R15                 CLEAR RETURN CODE REGISTER
         L     R14,12(R13)             RESTORE RETURN ADDRESS
         LM    R0,R12,20(R13)          RESTORE REGISTERS
         BR    R14                     RETURN
         EJECT
***********************************************************************
*                                                                     *
* END OF MAINLINE CODE.  SUBROUTINES FOLLOW.                          *
*                                                                     *
***********************************************************************
***********************************************************************
*                                                                     *
* PROCMSG:  PROCESSES MESSAGE BUFFERS ON THE PRIVATE QUEUE.           *
*                                                                     *
***********************************************************************
PROCMSG  EQU   *
         ST    R14,RPROCMSG            SAVE RETURN ADDRESS
         B     PROCMSG3                CONTINUE
***********************************************************************
*                                                                     *
* IF SUBTASK IS TERMINATING THEN SKIP WTO AND FREE THE MESSAGE BUFFER *
*                                                                     *
***********************************************************************
PROCMSG1 EQU   *
         TM    TVBIND1,TVBTERM         IS SUBTASK TERMINATING?
         BO    PROCMSG2                YES, ONLY FREE BUFFERS
***********************************************************************
*                                                                     *
* PROCESS THE MESSAGE BUFFER HERE                                     *
*                                                                     *
***********************************************************************
         L     R3,TVBMPRIQ             GET ADDRESS OF 1ST BUFFER
         MVC   MESSAGE(L'MSGRCV),MSGRCV
         MVC   MESSAGE+7(8),TVBOPID   GET TASK ID
         LA    R1,L'MSGRCV            GET MESSAGE ADDRESS
         BAL   R14,SENDMSG            DISPLAY MESSAGE
***********************************************************************
*                                                                     *
* DEQUEUE MESSAGE                                                     *
*                                                                     *
***********************************************************************
PROCMSG2 EQU   *
         USING BUFHDR,R3               R3 IS BASE FOR BUFFER HEADER
         L     R3,TVBMPRIQ             POINT TO FIRST BUFFER
         L     R1,HDRNEXTM-BUFHDR(,R3) ANCHOR THE NEXT MESSAGE
         ST    R1,TVBMPRIQ             MAKE IT THE FIRST BUFFER
```

Figure 14 (Part 5 of 11). Template for an Optional Task

```
***********************************************************************
*                                                                     *
* FREE THE MESSAGE BUFFER                                             *
*                                                                     *
***********************************************************************
         LH    R4,HDRBLENG-BUFHDR(,R3) GET THE BUFFER LENGTH
         DSIFRE R,A=(R3),LV=(R4),SP=0  FREE THE BUFFER
         LTR   R15,R15                 SUCCESSFUL?
         BZ    PROCMSG3                YES, CONTINUE
         MVC   MESSAGE(L'FRMFAIL),FRMFAIL
         MVC   MESSAGE+7(8),TVBOPID   GET TASK ID
         LA    R1,L'FRMFAIL           GET MESSAGE ADDRESS
         BAL   R14,SENDMSG            DISPLAY MESSAGE
***********************************************************************
*                                                                     *
* CHECK FOR MORE MESSAGE BUFFER                                       *
*                                                                     *
***********************************************************************
PROCMSG3 EQU   *
         OC    TVBMPRIQ,TVBMPRIQ       ANYTHING LEFT?
         BNZ   PROCMSG1                YES, KEEP PROCESSING
***********************************************************************
*                                                                     *
* EXIT MESSAGE PROCESSING                                             *
*                                                                     *
***********************************************************************
PROCMSGX EQU   *
         L     R14,RPROCMSG            RESTORE RETURN ADDRESS
         BR    R14                     RETURN
         DROP  R3
         EJECT
***********************************************************************
*                                                                     *
* THIS ROUTINE PERFORMS INITIALIZATION PROCESSING. A SWB CONTROL BLOCK*
* IS ALLOCATED AND INITIALIZED, THE TASK IS MARKED AS ACTIVE, AND     *
* A INITIALIZATION MESSAGE IS ISSUED.                                 *
*                                                                     *
***********************************************************************
INITRTN  EQU   *
***********************************************************************
*                                                                     *
* CLEAR STORAGE                                                       *
*                                                                     *
***********************************************************************
         XC    TVBUFLD,TVBUFLD         CLEAR STORAGE POINTER
         XC    TIBNDATD,TIBNDATD       CLEAR TIB WORKAREA
         XC    TIBEDATD,TIBEDATD       CLEAR TIB WORKAREA
```

Figure 14 (Part 6 of 11). Template for an Optional Task

```
***********************************************************************
*                                                                     *
* GET A 4096 BYTE WORK AREA.                                          *
*                                                                     *
***********************************************************************
         ST    R14,RINITRTN            SAVE RETURN ADDRESS
         DSIGET LV=4096,A=TVBUFLD,CLEAR=YES  GET 4K FOR WORKAREA
         LTR   R15,R15                 DID I GET THE STORAGE?
         BZ    INITRTN0                YES, CONTINUE
         MVC   MESSAGE(L'GETUFAIL),GETUFAIL
         MVC   MESSAGE+7(8),TVBOPID   GET TASK ID
         LA    R1,L'GETUFAIL          GET MESSAGE ADDRESS
         BAL   R14,SENDMSG            DISPLAY MESSAGE
         OI    TVBIND1,TVBTERM         SET TERMINATION FLAG
         B     INITRTNX                EXIT
***********************************************************************
*                                                                     *
* ISSUE DSILCS TO ACQUIRE A SERVICE WORK BLOCK (SWB).                 *
*                                                                     *
***********************************************************************
INITRTN0 EQU   *
         DSILCS CBADDR=SWBADDR,SWB=GET GET A SWB CONTROL BLOCK
         LTR   R15,R15                 DID I GET A SWB?
         BZ    INITRTN1                YES, GO INITIALIZE SWB
         MVC   MESSAGE(L'SWBFAIL),SWBFAIL
         MVC   MESSAGE+7(8),TVBOPID   GET TASK ID
         LA    R1,L'SWBFAIL           GET MESSAGE ADDRESS
         BAL   R14,SENDMSG            DISPLAY MESSAGE
         OI    TVBIND1,TVBTERM         SET TERMINATION FLAG
         B     INITRTNX                EXIT
***********************************************************************
*                                                                     *
* ENQ ON TVB CHAIN                                                    *
*                                                                     *
***********************************************************************
INITRTN1 EQU   *
         USING DSISWB,R1               R1 IS BASE FOR SWB
         L     R1,SWBADDR              GET ADDRESS OF SWB
         ST    R9,SWBTIB               PUT TIB ADDRESS IN SWB
         DROP  R1
         MVC   ENQDEQ(ENQDEQL),LENQDEQ INITIALIZE ENQDEQ LIST
         ENQ   (MVTNCCFQ,MVTTVBRN,E,18,STEP),MF=(E,ENQDEQ)
         LTR   R15,R15                 ENQ OKAY?
         BZ    INITRTN2                YES, CONTINUE
         MVC   MESSAGE(L'ENQFAIL),ENQFAIL
         MVC   MESSAGE+7(8),TVBOPID   GET TASK ID
         LA    R1,L'ENQFAIL           GET MESSAGE ADDRESS
         BAL   R14,SENDMSG            DISPLAY MESSAGE
         OI    TVBIND1,TVBTERM         SET TERMINATION FLAG
         B     INITRTNX                EXIT
***********************************************************************
*                                                                     *
* UPDATE TASK NAME AND SET TASK IS ACTIVE FLAG WHILE ENQ ACQUIRED     *
*                                                                     *
***********************************************************************
INITRTN2 EQU   *
         MVC   TVBOPID,TVBLUNAM        UPDATE NCCF TABLE WITH TASKID
         OI    TVBIND3,TVBACTV         INDICATE THAT I'M ACTIVE
```

Figure 14 (Part 7 of 11). Template for an Optional Task

```
***********************************************************************
*                                                                     *
* RELEASE ENQ ON TVB CHAIN. TASK IS NOW CONSIDERED ACTIVE.            *
*                                                                     *
***********************************************************************
         MVC   ENQDEQ(ENQDEQL),LENQDEQ INITIALIZE ENQDEQ LIST
         DEQ   (MVTNCCFQ,MVTTVBRN,18,STEP),MF=(E,ENQDEQ)
         LTR   R15,R15                 DEQ OKAY?
         BZ    INITRTN3                YES, CONTINUE
         MVC   MESSAGE(L'DEQFAIL),DEQFAIL
         MVC   MESSAGE+7(8),TVBOPID   GET TASK ID
         LA    R1,L'DEQFAIL           GET ADDRESS OF INITIALIZATION MSG
         BAL   R14,SENDMSG            GO DISPLAY INITIALIZATION MESSAGE
         OI    TVBIND1,TVBTERM         SET TERMINATION FLAG
         B     INITRTNX                EXIT
***********************************************************************
*                                                                     *
* DISPLAY INITIALIZATION MESSAGE                                      *
*                                                                     *
***********************************************************************
INITRTN3 EQU   *
         MVC   MESSAGE(L'INITMSG),INITMSG
         MVC   MESSAGE+7(8),TVBOPID   GET TASK ID
         LA    R1,L'INITMSG           GET ADDRESS OF INITIALIZATION MSG
         BAL   R14,SENDMSG            GO DISPLAY INITIALIZATION MESSAGE
***********************************************************************
*                                                                     *
* RETURN TO MAINLINE                                                  *
*                                                                     *
***********************************************************************
INITRTNX EQU   *
         L     R14,RINITRTN           RESTORE RETURN ADDRESS
         BR    R14                    EXIT
         EJECT
***********************************************************************
*                                                                     *
* THIS ROUTINE PERFORMS TERMINATION PROCESSING.  THE TASK IS MARKED   *
* AS INACTIVE, A TERMINATION MESSAGE IS ISSUED, AND THE SWB CONTROL   *
* BLOCK IS FREED.                                                     *
*                                                                     *
***********************************************************************
TERMRTN  EQU   *
         ST    R14,RTERMRTN            SAVE RETURN ADDRESS
```

Figure 14 (Part 8 of 11). Template for an Optional Task

```
***********************************************************************
*                                                                     *
* ENQUE ON TVB CHAIN PRIOR TO MARKING TASK INACTIVE                   *
*                                                                     *
***********************************************************************
         MVC   ENQDEQ(ENQDEQL),LENQDEQ INITIALIZE ENQDEQ LIST
         ENQ   (MVTNCCFQ,MVTTVBRN,E,18,STEP),MF=(E,ENQDEQ)
         LTR   R15,R15
         BZ    TERMRTN1
         MVC   MESSAGE(L'ENQFAIL),ENQFAIL
         MVC   MESSAGE+7(8),TVBOPID   GET TASK ID
         LA    R1,L'ENQFAIL           GET MESSAGE ADDRESS
         BAL   R14,SENDMSG            DISPLAY MESSAGE
TERMRTN1 EQU   *
         MVC   TVBOPID,=8XL1'40'      MAKE THIS TASK INACTIVE
         NI    TVBIND3,TVBACTV        MARK TASK AS INACTIVE
         MVC   ENQDEQ(ENQDEQL),LENQDEQ INITIALIZE ENQDEQ LIST
         DEQ   (MVTNCCFQ,MVTTVBRN,18,STEP),MF=(E,ENQDEQ)
         LTR   R15,R15                DEQ OKAY?
         BZ    TERMRTN2               YES, CONTINUE
         MVC   MESSAGE(L'DEQFAIL),DEQFAIL
         MVC   MESSAGE+7(8),TVBOPID   GET TASK ID
         LA    R1,L'DEQFAIL           GET MESSAGE ADDRESS
         BAL   R14,SENDMSG            DISPLAY MESSAGE
TERMRTN2 EQU   *
         OC    SWBADDR,SWBADDR        IS THERE AN SWB?
         BZ    TERMRTNX               NO, EXIT
         MVC   MESSAGE(L'TERMMSG),TERMMSG
         MVC   MESSAGE+7(8),TVBLUNAM  GET TASK ID
         LA    R1,L'TERMMSG           GET ADDRESS OF TERMINATION MSG
         BAL   R14,SENDMSG            GO DISPLAY TERMINATION MESSAGE
***********************************************************************
*                                                                     *
* FREE ACQUIRED SWB                                                   *
*                                                                     *
***********************************************************************
         DSILCS CBADDR=SWBADDR,SWB=FREE
         XC    SWBADDR,SWBADDR        CLEAR SWB ADDRESS
         LTR   R15,R15                WAS SWB FREED?
         BZ    TERMRTNX               YES, EXIT
         MVC   MESSAGE(L'SWBFFAIL),SWBFFAIL
         MVC   MESSAGE+7(8),TVBOPID   GET TASK ID
         LA    R1,L'SWBFFAIL          GET MESSAGE ADDRESS
         BAL   R14,SENDMSG            DISPLAY MESSAGE
```

Figure 14 (Part 9 of 11). Template for an Optional Task

```
***********************************************************************
*                                                                     *
* FREE ALL QUEUED STORAGE AND RETURN                                  *
*                                                                     *
***********************************************************************
TERMRTNX EQU   *
         L     R14,RTERMRTN            RESTORE RETURN ADDRESS
         BR    R14                     RETURN
         EJECT
***********************************************************************
*                                                                     *
* THIS ROUTINE SENDS A MESSAGE TO THE OPERATOR THAT STARTED THIS      *
* TASK.  IF THAT FAILS, THEN THE MESSAGE IS SENT TO THE AUTHORIZED    *
* RECEIVER.                                                           *
*                                                                     *
***********************************************************************
SENDMSG  EQU   *
         ST    R14,RSENDMSG              SAVE RETURN ADDRESS
         USING BUFHDR,R2                 R2 IS BASE FOR BUFFER HEADER
         LA    R2,BUFFER                 GET ADDRESS OF BUFFER
         STH   R1,HDRMLENG               PUT MESSAGE LENGTH IN BUFFER
         LA    R1,BUFHDRND-BUFHDR        GET OFFSET TO MSG TEXT
         STH   R1,HDRTDISP               MOVE MESSAGE OFFSET TO BUFFER
         AH    R1,HDRMLENG               MSG LEN+HDRTDISP
         STH   R1,HDRBLENG               MOVE BUFFER LENGTH TO BUFFER
         MVC   HDRDOMID(8),MVTCURAN      MOVE DOMAIN ID TO BUFFER
         MVI   HDRMTYPE,HDRTYPEU         INDICATE A USER MESSAGE
         DSIDATIM AREA=DATETIME,FORMAT=BINARY
         L     R1,DATETIME+4             GET TIME
         ST    R1,HDRTSTMP               PUT TIME IN BUFFER
         L     R1,SWBADDR                GET SWB ADDRESS
         DSIMQS SWB=(R1),BFR=(R2),TASKID=TIBMSGNM  SEND TO STARTER
         LTR R15,R15                     DID IT WORK?
         BZ    SENDMSGX
         DSIMQS SWB=(R1),BFR=(R2),AUTHRCV=YES
SENDMSGX EQU   *
         L     R14,RSENDMSG              RESTORE RETURN ADDRESS
         BR    R14                       RETURN
***********************************************************************
*                                                                     *
* END OF CODE.                                                        *
*                                                                     *
***********************************************************************
         EJECT
         LTORG
```

Figure 14 (Part 10 of 11). Template for an Optional Task

```
***********************************************************************
*                                                                     *
* THESE MESSAGES COULD BE SET UP IN A MESSAGE DEFINITION MODULE       *
* OR IN A MESSAGE DISK MEMBER.  SEE ABLDMSG FOR AN EXAMPLE OF USING   *
* DSIMBS TO CONSTRUCT MESSAGES.                                       *
*                                                                     *
***********************************************************************
INITMSG  DC    C'USR001 XXXXXXXX : TASK IS READY AND WAITING FOR WORK'
TERMMSG  DC    C'USR999 XXXXXXXX : TASK IS TERMINATED'
FRUFAIL  DC    C'USR002 XXXXXXXX : DSIFRE FAILED FOR USER STORAGE'
FRQFAIL  DC    C'USR003 XXXXXXXX : DSIFRE FAILED FOR QUEUED STORAGE'
MSGRCV   DC    C'USR004 XXXXXXXX : MESSAGE RECEIVED'
FRMFAIL  DC    C'USR005 XXXXXXXX : DSIFRE FAILED FOR MQS BUFFER'
GETUFAIL DC    C'USR006 XXXXXXXX : DSIGET FAILED FOR USER STORAGE'
SWBFAIL  DC    C'USR007 XXXXXXXX : DSILCS FAILED TRYING TO GET A SWB'
ENQFAIL  DC    C'USR008 XXXXXXXX : ENQ ERROR'
DEQFAIL  DC    C'USR009 XXXXXXXX : DEQ ERROR'
SWBFFAIL DC    C'USR010 XXXXXXXX : DSILCS FAILED TRYING TO FREE SWB'
LENQDEQ  ENQ   (,,,,),MF=L
ENQDEQL  EQU   *-LENQDEQ
         EJECT
DSITIB   DSECT
         ORG   TIBNDATD
         DS    0XL256
DATETIME DS    D
ECBLIST  DS    2F
RPROCMSG DS    F
RINITRTN DS    F
RTERMRTN DS    F
RSENDMSG DS    F
SWBADDR  DS    F
ENQDEQ   DS    CL(ENQDEQL)
         ORG   TIBEDATD
         DS    0XL256
BUFFER   DS    0F
         DS    XL(BUFHDRND-BUFHDR)
MESSAGE  DS    CL80
         END   AOPTTSK
```

Figure 14 (Part 11 of 11). Template for an Optional Task

Chapter 6

Chapter 6. Writing REXX User Functions

Introduction

This chapter explains how to add additional functions or expand or replace those REXX functions that already exist in the NetView/REXX environment. The process for MVS/XA is very similar, but for more specific MVS/XA information, refer to the *REXX Reference*.

Both the VM and the MVS/XA environments use the DSIRXEBS macro described in Chapter 8 to obtain storage for an evaluation block.

Overview of User-Written Functions

External functions can be written that allow you to extend the capabilities of the REXX language. You can write functions that supplement either the built-in functions or the functions that are provided. You can also write a function that will be used in place of a function already provided. For example, if you want a new substring function that performs differently from the SUBSTR built-in function, you can write your own substring function and name it STRING. Users at your installation can then use the STRING function in their REXX command lists.

NetView supports three types of function package directories. Basically, there are no differences between the three types. They are as follows:

- DSIRXUPD - User packages, which are function packages that an individual user may write to replace or supplement certain system-provided functions. When the function packages are searched, the user packages are searched before the local and system packages.
- DSIRXLPD - Local packages, which are function packages that a system support group or application group may write. Local packages may contain functions that are available to a specific group of users or to the entire installation. Local packages are searched after the user packages and before the system packages.
- System packages, which are function packages that have been written by NetView. System packages are searched after any user and local packages.

To provide new functions or change existing functions, there are several steps you must perform. The steps are described and explained in more detail in the following topics.

Subroutines can also be included in the function package directories.

Interface to Functions

When your code gets control, the function gets a control block called the evaluation block (EVALBLOK). The function places the result into the evaluation block, which is returned to the language processor. The result in the evaluation block is used in the interpretation of the REXX instruction that contained the function.

To obtain an evaluation block, see the NetView DSIRXEBS macro description.

Entry Specifications

When the code for the function gets control, the contents of the registers are:

Register 0	TIB address for VM, Environment Block for MVS/XA (TIB address is in ENVBLOCK_USERFIELD).
Register 1	Address of the external function parameter list (EFPL)
Registers 2-12	Unpredictable
Register 13	Address of a register save area
Register 14	Return address
Register 15	Entry point address

External Function Parameter List

When the function gets control, register 1 points to the external function parameter list, which is described in Table 4. The mapping macro IRXEFPL for the external function parameter list is provided by TSO/E.

Table 4. External Function Parameter List

Offset (Decimal)	Number of Bytes	Description
0	4	Reserved.
4	4	Reserved.
8	4	Reserved.
12	4	Reserved.
16	4	The address of the parsed argument list. Each argument is represented by an address/length pair. The argument list is terminated by X'FFFFFFFFFFFFFFFF'. Table 5 shows the format of the argument list.
20	4	The address of a fullword that contains the address of an evaluation block (EVALBLOK). The evaluation block is used to pass back the result of the function. Table 6 on page 111 describes the evaluation block.

Argument List

Table 5 shows the format of the parsed argument list the function receives at offset +16 (decimal). The mapping macro IRXARGTB for the argument list is provided by TSO/E.

Table 5. Format of the Argument List

Offset (Dec)	Number of Bytes	Field Name	Description
0	4	ARGSTRING_PTR	Address of argument 1
4	4	ARGSTRING_LENGTH	Length of argument 1
8	4	ARGSTRING_PTR	Address of argument 2
12	4	ARGSTRING_LENGTH	Length of argument 2
16	4	ARGSTRING_PTR	Address of argument 3
20	4	ARGSTRING_LENGTH	Length of argument 3
24	8	---	X'FFFFFFFFFFFFFFFF'

In the argument list, each argument consists of the address of the argument and its length. The argument list is terminated by X'FFFFFFFFFFFFFFFF'.

Evaluation Block

Before the function returns control to the language processor, the address of the evaluation block (EVALBLOK) is placed at offset +20 of the external function parameter list. The function computes the result and returns the result in the evaluation block.

The evaluation block consists of a header and data, in which you place the result from your function. Table 6 shows the format of the evaluation block.

The mapping macro IRXEVALB for the evaluation block is provided by TSO/E.

Table 6. Format of the Evaluation Block

Offset (Decimal)	Number of Bytes	Field Name	Description
0	4	EVPAD1	A fullword that contains X'00'. This field is reserved and is not used.
4	4	EVSIZE	Specifies the total size of the evaluation block in doublewords.
8	4	EVLEN	On entry, this field is set to X'80000000', which indicates no result is currently stored in the evaluation block. On return, specify the length of the result, in bytes, that your code is returning. The result is returned in the EVDATA field at offset +16.
12	4	EVPAD2	A fullword that contains X'00'. This field is reserved and is not used.
16	n	EVDATA	The field in which you place the result from the function or subroutine. The length of the field depends on the total size specified for the control block in the EVSIZE field. The total size of the EVDATA field is: `EVSIZE * 8 - 16`

The function must compute the result, move the result into the EVDATA field (at offset +16), and update the EVLEN field (at offset +8). If the initial evaluation block is too small to hold the complete result, you can use the DSIRXEBS macro to obtain a larger evaluation block. DSIRXEBS creates the new evaluation block and returns the address of the new block. Your code can then place the result in the new evaluation block. You must also change the parameter at offset +20 in the parameter list to point to the new evaluation block. If DSIRXEBS was used to get the initial evaluation block, DSIRXEBS will release the evaluation block if its address is provided when obtaining the larger evaluation block.

Functions must return a result. Subroutines are not required to return a result.

Directory for Function Packages

After writing the code for the function, you must create an entry in one of the directories. You need a directory entry for each individual function package you want defined.

A function package directory is contained in a load module and can be tailored to individual users or local groups. The name of the entry point at the beginning of the directory is the function package directory name. The name of the directory is specified only on the CSECT. In addition to the name of the entry point, the function package directories define each entry point for the individual functions that are part of the function package. The directories consist of two parts: a header followed by individual entries for each function included in the function package. Table 7 shows the format of the directory header. Table 8 illustrates the rows of entries in the function package directory.

Table 7. Format of the Function Package Directory Header

Offset (Decimal)	**Number of Bytes**	**Description**
0	8	An eight-byte character field that defines the directory. This is the name of the directory. For example, you can specify DSIRXUFP, which is one of the "dummy" function package names that is provided. The name must be in uppercase and left justified.
8	4	Specifies the length, in bytes, of the header. This is the offset from the beginning of the header to the first entry in the directory. This must be a fullword binary number equivalent to decimal 24.
12	4	The number of functions defined in the function package (the number of rows in the directory). The format is a fullword binary number.
16	4	Reserved
20	4	Specifies the length, in bytes, of an entry in the directory (length of a row). This must be a fullword binary number equivalent to decimal 32.

At offset +0 in the header, specify the name of the function package directory. Two "dummy" function package directory names are provided:

- DSIRXUFP for a user function package
- DSIRXLFP for a local function package

Format of Entries in the Directory

Table 8 shows two rows (two entries) in a function package directory. The first entry starts immediately after the directory header. Each entry defines a function or subroutine in the function package. The individual fields are described following the table.

Table 8 (Page 1 of 2). Format of Entries in Function Package Directory

Offset (Decimal)	**Number of Bytes**	**Field Name**	**Description**
0	8	FUNC-NAME	The name of the first function or subroutine (entry) in the directory.

Table 8 (Page 2 of 2). Format of Entries in Function Package Directory

Offset (Decimal)	Number of Bytes	Field Name	Description
8	4	ADDRESS	The address of the entry point of the function or subroutine (for the first entry).
12	4	—	Reserved.
16	8	SYS-NAME Reserved in VM.	The name of the entry point in a load module that corresponds to the function or subroutine (for the first entry). Not used by VM.
24	8	SYS-DD Reserved in VM	The ddname from which the function or subroutine is loaded. Not used by VM.
32	8	FUNC-NAME	The name of the second function or subroutine (entry) in the directory.
40	4	ADDRESS	The address of the entry point of the function or subroutine (for the second entry).
44	4	—	Reserved.
48	8	SYS-NAME Reserved in VM	The name of the entry point in a load module that corresponds to the function or subroutine (for the second entry). Not used by VM.
56	8	SYS-DD Reserved in VM	The ddname from which the function or subroutine is loaded. Not used by VM.

The following describes each entry (row) in the directory.

FUNC-NAME
The eight-character name of the external function or subroutine. This is the name that is used in the REXX command list. The name must be in uppercase and left justified.

If this field is blank, the entry is ignored.

ADDRESS
A four-byte field that contains the address, in storage, of the entry point of the function or subroutine. This address is used only if the code has already been loaded.

If the address is 0, the *sys-name* and, optionally, the *sys-dd* fields are used. A *LOAD* will be issued for *sys-name* from the DD *sys-dd.*

If the address is specified, the *sys-name* and *sys-dd* fields for the entry are ignored.

For VM, functions and subroutines must be loaded with NetView.

Reserved
A four byte field that is reserved.

SYS-NAME
An eight-character name of the entry point in a load module that corresponds to the function to be called for the *func-name.* The name must be in uppercase and left justified.

If the address is specified, this field can be blank. If an address of 0 is specified and this field is blank, the entry is ignored.

Not used by VM. See "**ADDRESS**" above.

SYS-DD

An eight-character name of the DD from which the function or subroutine is loaded. The name must be in uppercase and left justified. If the address is 0 and this field is blank, the module is loaded from the link list.

Not used by VM. See "**ADDRESS**" above.

Example of a User Function Directory

Figure 15 shows an example of a user function directory. The example is explained following the figure.

```
DSIRXUFP CSECT
         DC    CL8'DSIRXUFP'        String identifying directory
         DC    FL4'24'              Length of header
         DC    FL4'4'               Number of rows in directory
         DC    FL4'0'               Word of zeros
         DC    FL4'32'              Length of directory entry
*                                   Start of definition of first entry
         DC    CL8'MYF1    '        Name used in command list
         DC    FL4'0'               Address of preloaded code
         DC    FL4'0'               Reserved field
         DC    CL8'ABCFUN1 '        Name of entry point
         DC    CL8'        '        Reserved
*                                   Start of definition of second entry
         DC    CL8'MYF2    '        Name used in commmand list
         DC    FL4'0'               Address of preloaded code
         DC    FL4'0'               Reserved field
         DC    CL8'ABCFUN2 '        Name of entry point
         DC    CL8'        '        Reserved
*                                   Start of definition of third entry
         DC    CL8'MYS3    '        Name used in command list
         DC    AL4(ABCSUB3)         Address of preloaded code
         DC    FL4'0'               Reserved field
         DC    CL8'ABCFUN3 '        Name of entry point
         DC    CL8'        '        Reserved
*                                   Start of definition of fourth entry
         DC    CL8'MYF4    '        Name used in command list
         DC    VL4(ABCFUNC4)        Address of preloaded code
         DC    FL4'0'               Reserved field
         DC    CL8'        '        Name of entry point
         DC    CL8'        '        Reserved
         SPACE 2

         END  DSIRXUFP
```

Figure 15. Example of a Function Package Directory

In Figure 15, the name of the function package directory is DSIRXUFP, which is one of the "dummy" function package directory names provided. Four entries are defined in this function package:

- MYF1, which is an external function
- MYF2, which is an external function
- MYS3, which is a subroutine

- MYF4, which is an external function

For MVS/XA*:* If the external function MYF1 is called in a REXX command list, the load module with entry point ABCFUN1 is loaded from DD FUNCTDD1. If MYF2 is called in a REXX command list, the load module with entry point ABCFUN2 is loaded from the linklist because the *sys-dd* field is blank.

For VM*:* If the address is zero 0, processing will continue as if the entry was not found.

The load modules for MYS3 and MYF4 have been preloaded. The MYS3 subroutine and the MYF4 function have been assembled in a different object module but have been link edited as part of the same load module as the directory. The assembler, linkage editor, and loader have resolved the addresses.

When NetView is initialized, the load modules containing the function package directories for the environment are automatically loaded. The modules for the external functions and subroutines are loaded when a REXX command list calls the function or subroutine. All modules that are loaded remain loaded until the last REXX command list executing under the task under which the modules were loaded finishes executing.

Chapter 7

Chapter 7. Control Block Reference

This chapter describes control block fields related to customization interfaces. The other fields which may appear in control block mapping macros are considered the NetView program's internal use control information and are not to be used as a programming interface.

NetView control blocks and buffers passed to user exits and command processors are intended for **READ ONLY** use and should not be altered, with the exception of fields specifically designed as user fields, such as TVBUFLD, CWBSAVEA, and CWBADATD.

ART — Authorization and Routing Table

The authorization and routing table (ART) is a mapping consisting of multiple entries. Each entry includes the name of a VTAM resource as defined to NetView and indicators to define those spans for which the resource name is defined.

The mapping of the authorization and routing table is as follows:

Field Name	Length	Description
ARTCBH	8	Control block header.
ARTTABLE	*	Multiple entries.
ARTENTRY		Format of each DSIART entry.
ARTNAME	8	Resource name.
ARTIND	1	Indicator flags X'80' - This entry is active.
ARTSDEF	*	Variable length bit string, where each bit corresponds to a relative entry in DSISNT. If a bit is on (1), the resource is in the SPAN at the relative position in the SNT.

BUFHDR — Buffer Header

All message and command buffers must have an initialized buffer header (BUFHDR) preceding the buffer text. BUFHDR describes the buffer's size and usage, as well as the origin of the message or command. The items marked with an asterisk (*) in Figure 16 are the BUFHDR fields you must initialize.

BUFHDR is a separate DSECT contained in the task information block (TIB). To obtain the BUFHDR DSECT, use macro DSICBS to include the TIB control block.

BUFHDR is the only NetView control block that does not have the DSI prefix in its name.

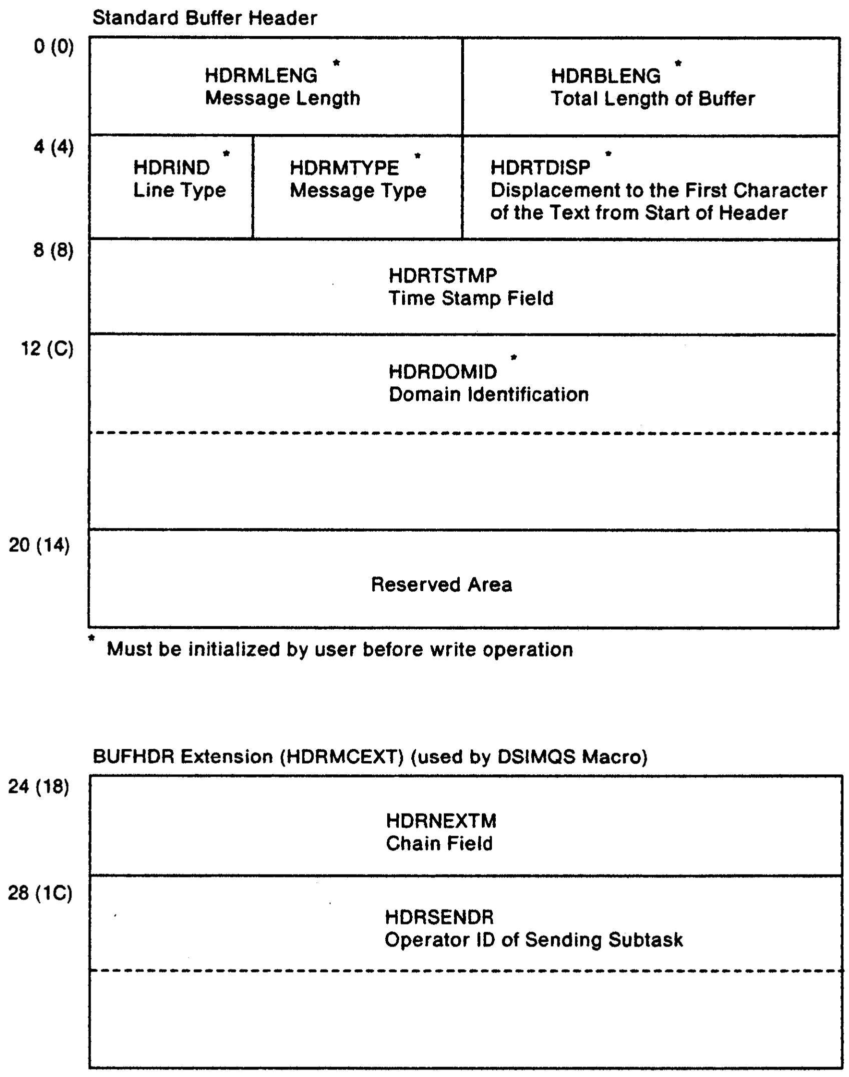


Figure 16. Buffer Header (BUFHDR)

An *initialized* BUFHDR has the fields set as follows:

Table 9 (Page 1 of 2). Control Block Fields

Field Name	Length	Description
BUFHDRND	0	Label at the end of BUFHDR for use in computing the length, in bytes, of BUFHDR.
HDRBLENG	2	The length, in bytes, of the entire buffer: header, text, and unused space. This length is used if the buffer is to be released with macro DSIFRE. It is a number between 0 and 32767 bytes.
HDRDOMID	8	The identifier of the domain that originated the message. This field is displayed and logged. The domain identifier under which a particular program is running is shown in the MVTCURAN field of the main vector table (MVT). It is recommended that the value of HDRDOMID equal the value of MVTCURAN. MVTCURAN is an eight-byte field that contains a DOMAINID (five or fewer bytes) and is padded with blanks (the rightmost three bytes are reserved).
HDRIND	1	This field is normally set to zero except when HDRMTYPE=HDRTYPEJ, HDRTYPEK, or HDRTYPEL. If HDRMTYPE=J, K, or L, then: HDRLNCTL TYPE = CONTROL LINE HDRLNLBL TYPE = LABEL LINE HDRLNDAT TYPE = DATA LINE HDRLNEND TYPE = DATA END LINE
HDRMCEXT	12	An extension that is appended to the BUFHDR when a buffer is transferred from one subtask to another. Macro DSIMQS BFRFLG=NO builds this extension when it creates a buffer copy for the destination task. If you want to pass the actual buffer with DSIMQS BFRFLG=YES, you must build the extension and initialize HDRSENDR.
HDRMLENG	2	The length, in bytes, of the text in the buffer.
HDRMSG	0	Label to indicate the place for text to begin if HDRMCEXT is present.
HDRMSGLN	0	Label at the end of HDRMCEXT for use in computing the length, in bytes, of BUFHDR + HDRMCEXT.
HDRMTYPE	1	Indicates the current usage of the buffer or the origin of the command. If the buffer is written using macro DSIPSS, this character is displayed and logged. For a list of values for HDRMTYPE, see Table 10 on page 123. You can use types B, I, L, T, and U only. Types B and T are used for commands only.
HDRNEXTM	4	MLWTO buffer chain pointer; points to the next buffer in the chain.
HDRSENDR	8	The originator's operator ID as found in the sender's TVBOPID field of the task vector block (TVB).
HDRTDISP	2	The offset from the start of the buffer header to the first byte of text.
HDRTEXT	0	Label to indicate the place for text to begin if HDRMCEXT is not present.

Table 9 (Page 2 of 2). Control Block Fields

Field Name	Length	Description
HDRTSTMP	4	The time that the buffer was created, in the packed decimal form X'*hhmmss* 0C', where: hh = the hour of the day, from 00 to 23; mm = the minutes of the hour, from 00 to 59; ss = the seconds of the minute, from 00 to 59; 0C = a packed decimal sign. To obtain values for this field, use DSIDATIM (see "DSIDATIM – Date and Time" on page 165).
Text		HDRTDISP indicates where the text starts. With a HDRMCEXT, text normally starts at HDRMSG. Without a HDRMCEXT, text normally starts at HDRTEXT. Text may start beyond these points, but not before them.

Values for HDRMTYPE Fields

For buffer header HDRMTYPE, the table below shows the values for the fields.

Table 10 (Page 1 of 3). Values for HDRMTYPE fields

Field Value	Meaning
HDRTYPEB (?)	Indicates a command or command list buffer that has display and logging suppressed. Not displayed on the operator's screen. Used to suppress display and logging of commands entered with a suppression character as defined in initialization member DSIDMN. Also used to suppress display and logging of command list statements that are preceded by this same suppression character.
HDRTYPEC (C)	Indicates a command or message from a command list. Changes to HDRTYPEB for suppressed command list statements. Changes to HDRTYPQC for quiet command.
HDRTYPED (!)	Indicates a message from an immediate command processor. Usually sent to the screen using DSIPSS TYPE=IMMED. When displayed in the immediate message area on the screen, the HDRMTYPE and DOMAIN name are not displayed. When received cross-domain, this type of message is in the normal output area, along with its domain name and type prefix. DSIPSS TYPE=IMMED does not enforce or set HDRTYPED.
HDRTYPEE (E)	Indicates a message from the SSI. This type is not used for title-line mode (MLWTO), system action, or WTOR messages. See also HDRTYPEK and HDRTYPEY for other forms of SSI messages.
HDRTYPEF (F)	Indicates a VSAM record. Not displayed on the operator's screen. Used within the data services task (DST).
HDRTYPEG (G)	Indicates a CNMI record. Not displayed on the operator's screen. Used within the data services task (DST).
HDRTYPEI (I)	Indicates an internal function request. This buffer is a formatted interface within and between tasks. The IFR contains a function number (IFRCODE) that determines the format and function of the buffer.
HDRTYPEJ (')	Indicates a title-line (MLWTO) message originating from NetView itself. These buffers must be in a sequence and include a description of control, label, data, and end designators. NetView treats these sequences of buffers as a single message for presentation and automation.
HDRTYPEK (")	Same as HDRTYPEJ, but for IBM non-NetView code.
HDRTYPEL (=)	Same as HDRTYPEJ, but for non-IBM written code.

Table 10 (Page 2 of 3). Values for HDRMTYPE fields

Field Value	Meaning
HDRTYPEM (M)	Indicates a message from the NetView message command processor.
HDRTYPEN (-)	Indicates a regular single buffer message from NetView.
HDRTYPEP (P)	Indicates a message or command from the PPT. This message type appears in the seventh character position of the domain name on the operator's screen, and is not the value in HDRMTYPE. This indicator appears in addition to the HDRMTYPE of the message itself.
HDRTYPEQ (Q)	Indicates a message from the VTAM POI that is a single-buffer unsolicited message. See also HDRTYPEV, HDRTYPEY, and HDRTYPEK for other VTAM POI messages. This message type is not set for messages from VTAM received on the SSI.
HDRTYPER (R)	Indicates that an operator entered the VTAM REPLY command in response to NetView WTOR number DSI802A. This message type is logged, but does not appear on NetView consoles.
HDRTYPES (S)	On some user exits interfaces, HDRMTYPE is set to HDRTYPES to indicate a swapped buffer.
HDRTYPET (*)	Indicates a command issued to NetView from a NetView terminal. This message type indicates that the buffer is a command rather than a message. IFRCODE=IFRCODCR is similar in that the buffer represents a command to be executed. Notice that IFRCODCR generally implies an internally formatted command, such as between OST and DST tasks. HDRTYPET generally implies a command buffer as if an operator had typed the command. IFRCODCR buffers can contain non-printable data. HDRTYPET buffers should contain no non-printable text.
HDRTYPEU (U)	Reserved for non-IBM users. Cannot be used for action messages (WTOR) or title-line (MLWTO) messages.
HDRTYPEV ( )	Indicates a message from the VTAM POI that is a single-buffer solicited message. See also HDRTYPEQ, HDRTYPEY, and HDRTYPEK for other VTAM POI messages. This message type is not set for messages from VTAM received on the SSI.
HDRTYPEW (+)	Indicates an IBM-written single line message. Similar to HDRTYPEN and HDRTYPEU.
HDRTYPEX (X)	Indicates a cross-domain (NNT to OST) command. Allows reverse-direction commands, since commands are normally routed from the OST to the NNT, for example, with the ROUTE command. Code running in a NNT task can issue DSIPSS TYPE=OUTPUT for a HDRTYPEX buffer, and the corresponding command will be executed in the OST that started the session with that NNT. This is useful for sending non-formatted (hexadecimal) data from the NNT to an OST for full-screen or other formatting. Limited to 256 bytes. Not displayed on the operator's screen.

Table 10 (Page 3 of 3). Values for HDRMTYPE fields

Field Value	Meaning
HDRTYPEY (>)	Indicates single-buffer action or WTOR. Can be a message from the SSI as well as from the VTAM POI. These messages remain on the screen until an action is taken or the reply is entered. The operator can delete these messages by overstriking the > character and pressing enter. The message disappears the next time the screen wraps over the text. When the HDRTYPEY flag is set and the IFRAUWQE flag is not set, NetView looks for a 3-character reply ID immediately preceding the message number in the message text. If the reply ID exists, the message is a VTAM WTOR. Otherwise, the message is treated as a held message (if IFRAUWQE is zero). If IFRAUWQE is set to 1, the IFRAUWQD data is checked to see if the WQE data indicates a WTOR or action message. If WTOR is indicated, a 2-character reply ID immediately precedes the message ID. If a reply ID exists, it is delimited from the message ID by one space.
HDRTYPEZ (Z)	Similar to HDRTYPEN, but specifically indicates a message from a data services task (DST).
HDRTYPE$ ($)	Indicates a non-displayable data message. Used for data transfer between high-level language command procedures.
HDRTYPLT (L)	Indicates a TRACE record. This message type is not displayed on the screen or in the NetView Logs.
HDRTYPQC ('32'X)	Indicates a command with all synchronous messages suppressed (performance option).
HDRTYPWT (W)	Indicates a message that matched a &WAIT and was displayed. The W appears in the message type field on the screen and in the logs but is not in the HDRMTYPE field in the buffer. The HDRMTYPE field in the buffer contains the original message type.
HDRTYPE1 (V)	Indicates PPOLOG echo of console operator command.
HDRTYPE2 (Y)	Indicates PPOLOG copy of console message, suppressed or unsuppressed.

Example of BUFHDR Usage

Macro DSIDKS uses the buffer header when reading a disk data set. You specify the blocking factor to block the disk data set. The disk services module DSIDRS prefixes the physical read buffer with a BUFHDR.

When the first record is requested, macro DSIDKS reads the first block. HDRTDISP is adjusted to indicate the first logical record. DSIDKS also sets HDRMLENG to reflect the logical record length. When DSIDKS is issued for the next logical record, HDRTDISP is adjusted to indicate the next logical record until the block is exhausted. DSIDKS reads another physical record; the process starts again from the first logical record in the block.

CBH — Control Block Header

The control block header (CBH) must precede all NetView control blocks, except the BUFHDR and the internal function request (IFR) control block. CBH identifies the length and type of control block that follows it, as well as the type of subtask the control block represents.

The relevant fields in the CBH are as follows:

Field Name	Length	Description
CBHID	1	A one-character identifier of the control block. CBHPDB is the identifier for a parse descriptor block (PDB).
CBHLENG	2	A halfword that contains the length of the control block. CBHLENG represents either the length that is preallocated or the length that is obtained by macro DSIGET. For example, a parse descriptor block (PDB) has both a fixed-size portion and a variable number of entries. For a PDB, CBHLENG contains the length of both parts.
CBHTYPE	1	The type of subtask that the control block represents. (The TIB and the TVB each contain this identifier.) The types are the primary POI task (PPT), the operator station task (OST), the NetView-NetView task (NNT), the hard-copy task (HCT), a data services task (DST), and the optional subtask (OPT). Macro DSILCS uses this field for managing control work blocks (CWBS) and service work blocks (SWBS). In all other cases, this byte is reserved. CBHHCT The identifier used for a TIB or a TVB belonging to an HCT task. CBHNNT The identifier used for a TIB or a TVB belonging to an NNT task. CBHOPTSK The identifier used for a TIB or a TVB belonging to an OPT task or a DST task. CBHOST The identifier used for a TIB or a TVB belonging to an OST task. CBHPPT The identifier used for a TIB or a TVB belonging to the PPT task.

CWB — Command Work Block

The command work block (CWB) contains the command processor parameters, a save area, and a work area. The relevant fields in the CWB are as follows:

Field Name	Length	Description
CWBCBH	4	A standard control block header.
CWBADATD	256	A work area for the command processor. If more storage is required, the command processor obtains it with macro DSIGET and releases it with macro DSIFRE. The command processor must free any storage it obtains.
CWBBUF	4	A pointer to a buffer containing a BUFHDR and the command text.
CWBDSRB	4	Used only by data services command processors (DSCPS). The data services task (DST) initializes this field with the address of the data services request block (DSRB). This field contains zero for all other command processor types.
CWBPARMS	12	A command processor parameter area. Its subfields are CWBBUF, CWBPDB, and CWBSWB.
CWBPDB	4	A pointer to a parse descriptor block (PDB), which is described under "PDB — Parse Descriptor Block" on page 146. The PDB contains parse information for the command pointed to by CWBBUF. If a special type of parse is required, the PDB may be reused by the command processor.
CWBRCODE	4	On resumption, an LRC receives a return code from a called command.
CWBSAVEA	72	A save area that may be utilized by the command processor.
CWBSWB	4	A pointer to a service work block (SWB) that the command processor may use or pass as a parameter to service macros or modules. Service macros build parameter lists in the SWB for the service modules. The SWB also contains a task information block (TIB) pointer, a parameter list, and a save area for use by the service routine. SWBS may be reused without re-initialization (service routines or macros need only the CBH and the TIB address).
CWBTIB	4	The address of the TIB for the subtask. The TIB and the task vector block (TVB) to which the TIB field TIBTVB is pointed contain all the information that relates to the subtask under which the command is running. This information contains the operator ID, the task type, and the task status.

DSB — Data Service Block

The data service block (DSB) contains information used by the disk read service routines called when the DSIDKS macro is issued. The following fields are used by the issuer of DSIDKS to access the records read:

Field Name	Length	Description
DSBBUFF	4	The address of the I/O buffer containing the record read from the data set member. The buffer contains a BUFHDR with HDRTDISP containing the offset of the requested logical record.
DSBREC	4	The address of the requested logical record read using the DSIDKS macro.

DSRB — Data Services Request Block

The data services request block (DSRB) contains information that a data services command processor (DSCP) needs to communicate with the DST. It also contains work space for the I/O routines. The relevant fields in the DSCP are as follows:

Field Name	Length	Description	
DSRBCBH	4	A standard control block header.	
DSRBCUSB	4	The address of a buffer used by the Communication Network Management (CNM) interface for unsolicited data. This field is used only when the DSRB function code (DSRBFNCD) indicates that unsolicited data has been received. The buffer contains a BUFHDR and the data length is in the HDRMLENG field of BUFHDR.	
DSRBFLG	1	The flag settings described below. The bits may be examined but not changed.	
		DSRBACTV = 1	There is an active transaction using this DSRB. A transaction is defined as a request from the time of its first arrival at the DSCP to the last exit of the DSCP. When a transaction ends, you can reassign the DSRB to another transaction.
		DSRBINUS = 1	Either the VSAM or the CNM interface service routine has an active request using this DSRB. DSRBINUS should not be on when DSRBACTV is off.
		DSRBTYPE = 1	The DSRB is used for unsolicited CNM data.
		DSRBTYPE = 0	The DSRB is used either for VSAM or CNM solicited data traffic.
DSRBFLGS	1	The flag settings described below. The bits may be examined but not changed.	
		DSRBCPMS = 1	The alert was generated from the distributed host.
		DSRBCPMS = 0	The alert came from the local CNMI.
DSRBFNCD	1	Indicator showing the reason that the command processor was called. The constants for DSRBFNCD are the following:	
		DSRBFNRM (1)	The first calling of the command processor, as the result of an IFRCODCR queued from another subtask using DSIMQS.
		DSRBFUNS (2)	The command processor was called to handle unsolicited CNM data.
		DSRBFSOL (3)	Solicited data was received from the CNM interface.
		DSRBFVSM (4)	A VSAM I/O request has completed.
DSRBINPT	4	The address of the CNM interface input buffer.	
DSRBOID	8	The ID of the operator that initiated the transaction.	
DSRBPRID	2	A halfword field that contains a correlation identifier for use by the CNM interface.	
DSRBRCMA	4	The return code for a completed request. It is set after the request is completed but before the DSCP is called again for request completion. This return code value is further explained by the minor return code (DSRBRCMI). See "DSCP Interface" on page 91 for a description of the return codes.	

Field Name	Length	Description
DSRBRCMI	4	The minor return code for a completed request. See DSRBRCMA.
DSRBTIB	4	The address of the DST task information block (TIB).
DSRBUBUF	4	The address of the original command that was sent to the DST. This field is unchanged during the data services transaction. This buffer contains a BUFHDR and the HDRMCEXT extension. It also has an X'0003' IFRCODE and HDRTYPEI. See "IFR — Internal Function Request" on page 133.
DSRBUSER	4	A field available for user purposes. If this field is used for a storage address, the DST does not free the storage. DSIGET Q=YES allocates storage. Storage may be freed as with any subtask that uses DSIFRE Q=YES. If storage is not freed, the storage remains allocated until the subtask terminates.
DSRBVACB	4	The address of the VSAM ACB for the DST
DSRBVDAD	4	The address of the VSAM I/O buffer with a standard BUFHDR. For GET requests, the BUFHDR HDRMLENG field indicates the length of the data read. HDRTDISP contains the offset to the data.
DSRBVECB	4	An event control block (ECB) for use by DST when requesting VSAM I/O.
DSRBVKEY	4	The address of the key in the DSRBVDAD buffer.
DSRBVKLN	2	The key length.
DSRBVRPL	4	The address of the VSAM RPL that was used for the I/O.
DSRBVRTP	1	An indicator of the type of request just completed: 1 - DSRVGET (VSAM GET) 2 - DSRVPUT (VSAM PUT) 3 - DSRVPNT (VSAM POINT) 4 - DSRVERS (VSAM ERASE) 5 - DSRVNRQ (VSAM ENDREQ)

ELB — External Logging Block

The external logging block maps the header information on the buffer passed to the XITXL exit.

Offset	Length	Name	Function
0	2	ELBLENG	Unsigned length of DSIELB
2	2	ELBRLENG	Unsigned length of record
4	1	ELBTYPE	Log type
5	3	ELBLOG	EBCDIC log type
8	4		Reserved by NetView
12			Start of record

Focal Point Transfer RU Header

The focal point transfer RU header is part of the CNM router support. All cross-domain unsolicited alert data is routed to the CNM router and the focal point transfer RU header carries management services information between distributed host and the focal point. The header is 44 bytes long and is followed by the NMVT. Some of the relevant fields are as follows:

Offset	Name	Length	Description
0	HDR LEN	2 bytes, binary	Length of the total RU (includes RU header and NMVT)
2	HDR ID	2 bytes	Alway X'1040'
4	Reserved	11 bytes	For NetView use only
15	DOMID LEN	1 byte, binary	Originator's domain ID length
16	DOMAIN ID	8 bytes, char	Originator's domain ID, padded with blanks
24	Reserved	20 bytes	For NetView use only
44	NMVT data		

IFR — Internal Function Request

The internal function request (IFR) is a formatted buffer that is transmitted to a subtask's message queue using macro DSIMQS. The relevant field in the IFR is as follows:

Field Name	Length	Description	
IFRCODE	2 bytes	Specified as one of the following:	
		IFRCODCR	The remainder of the buffer is a command to run.
		IFRCODUS	The buffer is user-defined and is passed to DSIEX13, the message receiver exit routine (applies only to an OST or NNT).
		IFRCODPN	The command is entered from a full-screen panel.
		IFRCODAI	Message automation internal function request (AIFR). (See "AIFR — Automation Internal Function Request" for the subfields of the AIFR.)

AIFR — Automation Internal Function Request

When IFRCODE=IFRCODAI, the fields described below are defined starting at the location with label IFRPARMS. For the automation IFR (AIFR) the value of HDRTDISP is always decimal 36, which includes a standard 24-byte BUFHDR and a 12-byte HDRMCEXT.

Note: The AIFR is 256 bytes long including the headers.

Field Name	Length or Mask	Subfield or EQU	Description
IFRAUIND	1 byte		Contains the primary AIFR control flags. These identify optional fields and describe routing and processing actions.
	1...	IFRAUWQE	See also IFRAUSSI. When this flag is *on*, fields in IFRAUWQD are set from the WQE data received from the system. This data is also forwarded to all NetView domains by NNTS. **Note:** If the IFRAUTBA buffer has HDRMTYPE=HDRTYPEY and the IFRAUWQE flag is *on*, the message must have a two-digit EBCDIC reply ID in front of the message number in the text.
	.1..	IFRAUCMD	Set by message table to indicate that an action is in the buffer referenced by IFRAUCMB. There is a separate AIFR structure for each command action for a particular message. **Note:** IFRAUCMD and IFRAUACT must not both be *on* at once.
	..1.	IFRAUACT	Indicates that message actions exist in the bits in IFRAUTA1. See IFRAUTA1 for a description of these. IFRAUCMD and IFRAUACT must not both be *on* at once. If both IFRAUCMD and IFRAUACT are zero, the buffer is subject to defaults and overrides. This bit must be *on* to specify force flags in IFRAUTA4.

Field Name	Length or Mask	Subfield or EQU	Description
	...1	IFRAUMTB	Used to control recursion and to prevent secondary receiver and copied messages from being processed by the message table. This bit can be used by DSIEX02A to prevent message table processing. IFRAUMTB is turned *off* when received from an NNT to allow automation of cross–domain messages. **Note:** Secondary receiver and copied messages are allowed to be automated when received as cross–domain messages.
	.... 1...	IFRAUPPT	Indicates that a message originated in the PPT. This bit replaces the "P" designation in byte 7 of HDRDOMID. The "P" is only displayed and logged as part of the domain name but is never in the buffer header.
	.... .1..	IFRAUXDM	Indicates that a message was received from an NNT. This bit can be used to determine if a message was from another domain.
	.... ..1.	IFRAUDOM	Indicates that this is a system delete message request. The system deletes messages by SMSGID values, by ASID and TCB address, or by ASID alone. This bit must be set only by NetView.
	1	IFRAUDFL	IFRAUDFL and IFRAUDF2 are valid when IFRAUDOM is *on*. IFRAUDFL=OFF and IFRAUDF2=OFF means delete by SMSGID. IFRAUDFL=ON and IFRAUDF2=OFF means delete by ASID. IFRAUDFL=ON and IFRAUDF2=ON means delete by ASID and TCB address.
IFRAUIN2	1 byte		The second byte of indicator flags.
	1...	IFRAUDF2	IFRAUDFL and IFRAUDF2 are valid when IFRAUDOM is *on*. IFRAUDFL=OFF and IFRAUDF2=OFF means delete by SMSGID. IFRAUDFL=ON and IFRAUDF2=OFF means delete by ASID. IFRAUDFL=ON and IFRAUDF2=ON means delete by ASID and TCB address.
	.1..	IFRAUEX2	Used to prevent calling DSIEX02A a second time. This bit is *on* when DSIEX02A is called to insure that DSIMQS of buffers from DSIEX02A will not cause re-drives. IFRAUEX2 is reset when received as a cross–domain message. This flag also prevents calls to DSIEX04 and DSIEX09.
	..1.	IFRAUPRI	Indicates that a message was routed using ASSIGN PRI. This is the primary copy of the message and is subject to automation. This bit replaces the indicator in HDRDOMID. This bit prevents ASSIGN COPY.
	...1	IFRAUSEC	Indicates that a message was routed using ASSIGN SEC. This is a secondary copy of the message and is not subject to automation. This bit replaces the indicator in HDRDOMID. IFRAUMTB is set *on* also. This bit prevents ASSIGN COPY.

Field Name	Length or Mask	Subfield or EQU	Description
	.... 1...	IFRAUCPY	Indicates that a message was routed using ASSIGN COPY. This is a secondary copy of the message and is not subject to automation. This bit replaces the indicator in HDRDOMID. IFRAUMTB is set *on* also. This bit prevents ASSIGN COPY.
	.... .1..	IFRAUAUT	Indicates that the buffer was sent using DSIMQS using authorized receiver routing. This bit replaces the indicator in HDRDOMID. This bit prevents ASSIGN COPY.
	.... ..1.	IFRAUDLD	Indicates that a message was received from a down-level (pre-NetView Release 2) domain. The AIFR indicators, HDRDOMID, and HDRSENDR are not reliable.
	1	IFRAUNSL	Indicates that the message will be routed using unsolicited receiver rules. This applies to unsolicited SSI traffic, DSIMQS AUTHRCV, and unsolicited VTAM messages.
IFRAUWID	4 bytes		The SMSGID value used for DOM purposes and to collect MLWTO buffers. This is a copy of WQE data.
IFRAUTCB	4 bytes		The job step TCB address for the issuer of the WTO. It is used as a correlation value and may not be an address in the current memory (or machine).
IFRAUASI	2 bytes		The address space ID for the issuer of the WTO. It is used as a correlation value and may not be an address space in this machine.
IFRAUSRB	2 bytes		Provided for the user. It will be initialized to zeros when created by NetView and will be copied into all copies of the original. It will not be changed.
IFRAUTBA	4 bytes		The pointer to data buffers. These are queued using HDRNEXTM to point to the next one. An entire MLWTO or title-line message is chained to this field. The buffers must have standard NetView buffer headers and message buffer header extensions.
IFRAUTBL	4 bytes		Points to the last buffer in the chain and is primarily intended for allowing buffers to be added to the end of a message without the need to scan the chain looking for the end.

Field Name	Length or Mask	Subfield or EQU	Description
IFRAUTA1	1 byte		IFRAUTA1 and IFRAUTA2 provide the message table processing values. They can be set by DSIEX02A but are subject to message table and OVERRIDE command changes. (See also IFRAUTA4 force flags.) 00.. .... HOLD - default 10.. .... HOLD - YES 01.. .... HOLD - NO 11.. .... Reserved ..xx .... Not used .... 00.. SYSLOG - default .... 10.. SYSLOG - YES .... 01.. SYSLOG - NO .... 11.. Reserved .... ..00 NETLOG - default .... ..10 NETLOG - YES .... ..01 NETLOG - NO .... ..11 Reserved
IFRAUTA2	1 byte		Option bits 00.. .... HCYLOG - default 10.. .... HCYLOG - YES 01.. .... HCYLOG - NO 11.. .... Reserved ..00 .... DISPLAY - default ..10 .... DISPLAY - YES ..01 .... DISPLAY - NO ..11 .... Reserved .... 00.. BEEP - default .... 10.. BEEP - YES .... 01.. BEEP - NO .... 11.. Reserved .... ..xx Not used
IFRAUTA3	1 byte		More display flags
	1...	IFRAUAOI	Indicates whether *all* or *one* routing is done as stated in the message table. A bit value of 1 indicates "ALL" while a bit value of 0 indicates "ONE". This flag is internal to message table processing and to the DSIEX16 interface.
	.1..	IFRAUPFU	Indicates that a message is full-line mode. This is similar to title-line processing, except that messages will start at the top of the screen. This bit is internal to NetView display management.
	..1.	IFRAUPFW	Indicates that a message is wide-line mode. This bit is internal to NetView display management.
	...1	IFRAUSSI	Indicates that a message was received on the SSI interface. This is to allow independent testing for WQE data from the indication that a message has been subject to system logging via WTO.

Field Name	Length or Mask	Subfield or EQU	Description
	.... 1...	IFRAUWAT	Indicates that a message has satisfied a wait in a command list. This bit indication replaces changing HDRMTYPE to HDRTYPWT, which obscured the HDRMTYPE value. DSIEX16 is driven with the message, and upon return NetView frees the buffer structures if all the display and log action indicators still say no-display, no-log, and suppress override options. The driving of DSIEX16 for wait–suppressed messages is intended for accounting purposes, although resetting the indicators would allow logging of wait-suppressed messages, for example.
	.... .1..	IFRAUX16	Indicates that DSIEX16 has been driven and prevents DSIEX16 from being driven again for this message. This bit is set to zero when received on a cross–domain session to allow processing in the new domain.
	.... ..1.	IFRAUVSE	Indicates that the message is in VSE format (Partition ID), (Reply ID), (Message ID). This bit is not reset when received cross–domain, allowing messages to retain their format integrity.
IFRAUTA4	1 byte		The following six flags force the actions specified in IFRAUTA1 and IFRAUTA2 to be in effect regardless of the OVERRIDE command options specified for this task. If the bit is 1, the override is ignored. If the related IFRAUTA1 or IFRAUTA2 flags are B'00', the action imposed by the DEFAULTS command or initial setting applies. The IFRAUFHD and IFRAUFBP when specified with non-zero values for the related IFRAUTA1 or IFRAUTA2 flags cause the BEEP or HOLD action to occur regardless of the initial settings or DEFAULTS command options. These flags may be set by NetView and are not reset when received cross domain so that DSIEX02A in the new domain can get a true picture of the original message. NetView sets IFRAUFSL, IFRAUFNL, and IFRAUFHL to prevent duplicate logging in multiple routed copies. When force flags are set before message automation occurs, either by NetView before DSIEX02A or by DSIEX02A, the automation table values for that option are ignored.
	1...	IFRAUFHD	Force HOLD actions
	.1..	IFRAUFSL	Force SYSLOG actions
	..1.	IFRAUFNL	Force NETLOG actions
	...1	IFRAUFHL	Force HARDCOPY actions
	.... 1...	IFRAUFDS	Force DISPLAY actions
	.... .1..	IFRAUFBP	Force BEEP actions

Field Name	Length or Mask	Subfield or EQU	Description
IFRAUCMB	4 bytes		Points to a buffer with HDRMTYPE=HDRTYPEI and IFRCODE=IFRCODCR used for command action. This buffer is built by the message table. Data buffers are sent on the IFRAUTBA chain with the command.
IFRAUPRS	4 bytes		Built and used only during message table processing.
IFRAUSRC	16 bytes		Provided for 16 bytes of user data. It is set to zeros when the AIFR is created and is copied from the original buffer without change.
IFRAUSDR	8 bytes		Used to retain the HDRSENDR value when sending AIFRS from one domain to another.
IFRAUTAF	8 bytes		Used to retain the TAF session name when replacing it with the domain ID in AIFR HDRDOMID.
IFRAURTL	4 bytes		The address of the route action list only used during calls to DSIEX16. The routing list is mapped by the IFRAURTB mapping. The IFRAUAOI bit determines whether the *first* or *all* active tasks in the list specified are sent the buffer.
IFRAURSV	20 bytes		Reserved for expansion. It should not be used for any purpose.
IFRAUWQD	120 bytes		Maps the WQE data that has been processed into a format independent of levels of MVS and JES.
IFRAUWHD	4 bytes		Eye catcher MSG or DOM.
IFRAUWF1	1 byte		Flags
	1...	IFRAUWFR	MLWTO first
	.1..	IFRAUWMD	MLWTO middle
	..1.	IFRAUWLS	MLWTO last
	...1	IFRAUWDO	DOM
	.... 1...	IFRAUWSI	Single message
	.... .1..	IFRAUWWR	WTOR
	.... ..1.	IFRAUWSP	Message suppressed
	1	IFRAUWBD	Broadcast to all
IFRAUWF2	1 byte		Flags 2
	1...	IFRAUWJN	Display job names
	.1..	IFRAUWST	Display status
	.... .1..	IFRAUWSS	Display session

Field Name	Length or Mask	Subfield or EQU	Description
IFRAUWF3	1 byte		Flags 3
	1...	IFRAUWTA	Control line
	.1..	IFRAUWTB	Label line
	..1.	IFRAUWTC	Data line
	...1	IFRAUWTD	End line
IFRAUWF4	1 byte		Flags 4
	1...	IFRAUWDT	DOM by ASID and TCB
	.1..	IFRAUWDA	DOM by ASID
IFRAUMCS	2 bytes		MCS flags
IFRAUWMA	1 byte		Area ID
IFRAUWDS	2 bytes		Descriptor codes
IFRAUWAS	2 bytes		ASID of issuer
IFRAUWJT	4 bytes		JSTCB of issuer
IFRAUWWI	4 bytes		SMSGID value
IFRAUWUC	4 bytes		Target console UCMID
IFRAUWSN	8 bytes		System name
IFRAUWRT	16 bytes		Route codes
IFRAUWTL	4 bytes		Text length
IFRAUWJU	8 bytes		Job number
IFRAUWJA	8 bytes		Job name
IFRAUWDF	1 byte		Message format

Note: The AIFR is fixed length at 256 bytes including the BUFHDR and HDRMCEXT. Do not expand it.

Automation Internal Function Request Routing List

IFRAURTB is the mapping of the route action list that is used during DSIEX16 processing. IFRAURTL is the address of a standard NetView buffer with a standard buffer header and a standard buffer header extension. The route list within this buffer contains a list of names who get identical copies of the IFRCODAI structure. Bit IFRAUAOI determines whether all active or the first active name receives a copy of the IFRCODAI. HDRMSG is a field within BUFHDR that is part of the DSITIB control block macro. This buffer must be addressed separately from the IFRCODAI buffer. HDRBLENG, HDRMLENG, and HDRTDISP are the only fields that must be initialized in this buffer header.

Field Name	Length	Description
IFRAURCC	2 bytes	A halfword count of the number of route names that follow. The HDRBLENG value must be sufficient to contain this number of names plus the buffer headers.
IFRAURCL	10 bytes	A variable size array of 10-character names. These must contain an operator ID of no more than 8 characters padded with blanks to a length of 10. Multiple 10 character fields continue here.

Usage Notes

Setting the message action flags in the buffer in DSIEX02A is equivalent to specifying the options in the message automation table. If the message is selected by the message automation table, any options specified by the table overlay the DSIEX02A values. The message is then evaluated after DSIEX02A and table processing against criteria specified by DEFAULTS and OVERRIDE commands to determine display and logging actions. IFRAUACT must be set to 1 if any ACTIONS are selected by DSIEX02A.

In the BUFHDR that precedes the IFR:

1. HDRMTYPE is specified as "I" (HDRMTYPE=X'C9'; symbol HDRTYPEI). Because an IFR is transferred by DSIMQS, the IFR contains a message command extension when it is received. The extension is optional in some cases but is recommended for consistency. If a command processor receives control with a command buffer and HDRMTYPE=HDRTYPEI, it is assumed that there is a command extension and an IFR.

2. HDRTDISP contains the displacement to the IFRCODE. For IFRCODCR and IFRCODUS, NetView modifies HDRTDISP and HDRMLENG so that all commands appear the same to the command processor. The command verb is followed by the parameters. The IFR section is logically removed.

LOGDS — NetView Log DSECT

The NetView log DSECT maps the record written to the network log.

The relevant fields in the LOGDS are as follows:

Field Name	Length	Description
LOGDATE	4	Date of record; format is 00YYDDDC.
LOGDISP	2	Record text displacement.
LOGDOMID	8	Domain ID of record originator.
LOGKEYDT	4	Same as LOGDATE.
LOGKEYTM	4	Time of record; format is HHMMSS0C, where HH is the hour, MM is the minute, SS is the second, and 0C is the symbol for packed decimal.
LOGLUNAM	8	Task name of record originator.
LOGMTYPE	1	Message type of record.
LOGOPID	8	Operator ID of record originator.
LOGSEQ#	4	Sequence number for VSAM key.
LOGTEXT		Text of Network log record.
LOGTIME	4	Same as LOGKEYTM.

MVT — Main Vector Table

The main vector table (MVT) is the main control block for information throughout NetView. It contains global information, such as the domain name, the status of NetView, and pointers to other tables and subtasks.

There is one MVT for each NetView. You can locate the MVT from a subtask through the TVBMVT, a pointer in the task vector block (TVB). A USING statement for the DSECT for the MVT must precede most macro invocations.

The relevant fields in the MVT are as follows:

Field Name	Length	Description
MVTARTLN	2	Length of each entry in DSIART.
MVTCBH	4	A standard control block header.
MVTCDSES	2	The number of OSTs in other domains that may have sessions at one time with this NetView. This is the number of TVBs created for NNTs in the TVB chain. This number is specified in the CDMNSESS definition statement.
MVTCLOSE	1	A flag bit that indicates that the CLOSE NORMAL command has been issued. When the bit is on, no more subtasks are attached and logons are not accepted.
MVTCURAP	9	The value from the NCCFID definition statement DOMAINID parameter, as follows: MVTCURAL 1-byte field that shows the length of the NCCFID DOMAINID (one to five characters). MVTCURAN 8-byte field that contains the NCCFID DOMAINID padded with blanks.
MVTDPRAD	4	Address of NetView subtask dispatcher (for compatibility with NetView Release 2 VSE only).
MVTDRTRY	2	The number of times an I/O operation is retried before it becomes a permanent error.
MVTGFMG1	4	A pointer to a Write-to-Operator parameter list containing the message, DSI124I STORAGE REQUEST FAILED FOR NCCF. This message may be used by any WTO macro with MF=E. No additional storage is required. The routing code is (2,11); the descriptor code is 11.
MVTGFMG2	4	A pointer to a Write-to-Operator parameter list containing the message, DSI125I CRITICAL STORAGE SHORTAGE FOR NCCF. This message may be used by any WTO macro with MF=E. No additional storage is required. The routing code is (2,11); the descriptor code is 11.
MVTGMSG	4	Pointer to a buffer containing the message, DSI073A COMMAND PROCESSOR UNABLE TO BUILD RESPONSE MESSAGE.
MVTMETH	1	Indicates that the access method is VTAM (V).
MVTMLGON	2	The number of times invalid logon information is processed before that terminal session ends. This number is specified in the MAXLOGON definition statement.
MVTMRC	2	The number of times an OST or PPT may ABEND and be reinstated. This number is specified in the MAXABEND definition statement.

Field Name	Length	Description
MVTNCCFQ	8	The QNAME value for macros ENQ and DEQ.
MVTSNT	4	Address of SPAN Name Table.
MVTSNTLN	2	Length of each entry in DSISNT
MVTSVL	4	The address of the service routine vector list (SVL) that contains the addresses of the service routines.
MVTTOD	8	The system time-of-day clock when NetView was started.
MVTTVB	4	The address of the first TVB in the TVB chain.
MVTTVBRN	18	The RNAME value for the TVB chain.
MVTUFLD	4	A user-defined field.
MVTVER	4	Contains a displayable identifier for the release of NetView under which your code is running.

MVT DSIEX16 Interface Data

For field name MVTAIDFT, the following defaults are set in DSIMVT and relate to DSIEX16.

Length or Mask	Start-up Value	Description
1 byte		Contains message process defaults.
`1... ....`	1	1 = Messages are allowed to be held on the screen. 0 = Messages are not allowed to be held on the screen.
`.x.. ....`		Reserved
`..0. ....`	0	0 = Messages other than NetView WTO and WTOR are not written to the system log. 1 = NetView messages are written to the system log.
`...1 ....`	1	1 = Messages are written to the NetView log if the NetView log is active. 0 = Messages are not written to the NetView log.
`.... 1...`	1	1 = Messages can be written to the hardcopy log if one is started for the operator. 0 = Messages are not written to the hardcopy log.
`.... .1..`	1	1 = Messages can be displayed on the NetView operator's station. 0 = Messages can not be displayed on the NetView operator's station.
`.... ..1.`	1	1 = Messages may beep on the NetView operator's station. 0 = Messages can not beep on the operator's station.
`.... ...x`		Reserved

Notes:

1. These settings are changed by using the DEFAULTS command.
2. The interpretation of the results of the settings is affected by the settings of AIFR flags for a message and by the OVERRIDE command settings currently in effect for each operator.
3. Some messages are not affected by any or all of the above.

OIT — Operator ID Table

The operator id table (OIT) is a mapping consisting of multiple entries. Each entry includes the name of the NetView operator ID and indicators relative to that operator ID. This table is used by the DSIOIS macro.

The mapping of the operator ID table is as follows:

Field Name	Length	Description
OITCBH	4	Control block header.
OITTABLE	*	Multiple entries.
OITENTRY		Format of each DSIOIT entry.
OITID	8	Operator ID.
OITIND	2	Indicator flags X'80' - operator is logged on to this NetView. X'40' - operator is in session with this NetView.

PDB — Parse Descriptor Block

The parse descriptor block (PDB) contains parse information for a command pointed to by CWBBUF in the CWB or by USERPDB in the USE. The PDB has no fixed length. The relevant fields in the PDB are as follows:

Field Name	Length	Description
PDBCBH	4	Standard control block header.
PDBBUFA	4	The address of the command buffer. CWBBUF also contains this address.
PDBCMDA	4	A pointer to the entry in the system command table (SCT) for the verb in the buffer that caused this command processor to be called. Macro DSIPAS (parameter alias services) and DSIKVS (KEYCLASS and VALCLASS lookup services) use this entry as a parameter.
PDBFLAGS	1	Indicator byte for command processors. **PDBIMMED** 1 = runs as an immediate command, 0 = runs as a regular command or a data services command.
PDBNOENT	2	The number of syntactical element entries in the PDB, including the verb and all parameters.
PDBTABLE	0	Label to indicate the beginning of PDB entries.
PDBENTRY	4	Each syntactical element creates one entry in this portion of the PDB. The verb is always the first entry. Each entry contains the length, the delimiter, and the offset from the beginning of the buffer, as specified in the following three fields: **PDBDISP** A 2-byte field specifying the offset from the start of the buffer to the first character of the *n*th syntactical element. For example, element addr(n) = PDBBUFA + PDBDISP(n). **PDBLENG** A 1-byte field specifying the length of the particular syntactical element. This field does not include the length of the delimiter. When two delimiters other than blanks occur sequentially, the length is zero. Also, when two delimiters are separated by a blank or blanks, the value of the length is zero. The offset is set to point to the second delimiter. **PDBTYPE** A 1-byte field containing the delimiter character that separates this element from the next one. The standard parsing delimiters are blank, comma, period, and equal sign. The end of the record is treated as if it were delimited by a blank. Multiple blanks are treated as one blank. Blanks preceding a syntactical element are ignored. For example, " verb parameter1,parameter2" creates the following: 1. An entry for the verb (preceding blanks are ignored) 2. An entry for *parameter1*, delimited by a comma 3. An entry for *parameter2*, delimited by a blank. **PDBENTND** A label at the end of PDBENTRY for use in computing the length of PDBENTRY.

SCE — System Command Entry

The system command entry (SCE) contains information about a command. Macro DSICES returns the SCE of a verb or command processor. The address of the SCE is also input to macro DSIPAS.

The relevant fields in the SCE are as follows:

Field Name	Length	Description
SCECADDR	4	For MVS/XA, this is the address of the linkage assist routine for command processors, DSICMDLD. For operating systems other than MVS/XA, this is the address of the command processor's entry point.
SCELNAME	8	The load module or phase name of the command processor to be called for the verb.
SCERCADR	4	The address of the command processor's entry point for MVS/XA. For operating systems other than MVS/XA, this field has no meaning.
SCEVERB	8	The command verb, left-justified and padded with blanks.

SCT — System Command Table

The system command table (SCT) contains a system command entry (SCE) for each command defined to NetView in the CMDMDL definition statement. There are no user fields here; however, you must include this control block mapping to get a mapping of the SCE.

SNT — Span Name Table

The span name table (SNT) is a mapping consisting of multiple entries. Each entry includes the name of the span of definition and indicators to define those operators that may be controlling the span. This table is used by the DSISSS macro.

The mapping of the span name table is as follows:

Field Name	Length	Description
SNTCBH	4	Control block header.
SNTTABLE	*	Multiple entries.
SNTENTRY		Format of each DSISNT entry.
SNTNAME	8	Span name.
SNTOPDEF	*	Variable length bit string, where each bit corresponds to a relative entry in the Operator ID table. If a bit is on (1), the corresponding operator may control this span.

SVL — Service Routine Vector List

The service routine vector list (SVL) is used by NetView macros to call the requested service routine. The SVL contains the addresses of all service routines, except for presentation service routines. There are no user fields here; however, you must include this control block mapping to use NetView services.

SWB — Service Work Block

The service work block (SWB) contains the parameter list for most service facilities used in user-written code.

The relevant fields in SWB are as follows:

Field Name	Length	Description
SWBADATD	256	Automatic work area to be used for reentrant variable definition.
SWBLRCPL	*	Mapping DSECT for area where caller builds parameter lists for DSIPUSH, DSIPOP,and DSIFIND.
SWBPLIST	256	Parameter list area.
SWBSAVEA	72	A standard save area.
SWBTIB	4	Pointer to the caller's TIB.
MQSENTTO	8	Operator ID of the operator to whom the message was sent. Returned to the issuer if DSIMQS is issued with a LIST of type 1ST

For a description of SWBLRCPL fields, see"DSIPUSH — Establish Long Running Command" on page 202, "DSIPOP — Remove Long Running Command" on page 190, and "DSIFIND — Find Long Running Command Storage" on page 169.

TIB — Task Information Block

The task information block (TIB) keeps information about an attached subtask. TIB is acquired and freed by the main task.

The TIB fields described below apply to optional subtasks:

Field Name	Length	Description
TIBCBH	4	A standard control block header. The CBHTYPE field contains the same information as the CBHTYPE field in the TVB.
TIBACB	4	A pointer to a VTAM ACB for NetView subtasks. This field may be defined by the subtask.
TIBAPID	9	The VTAM application program name for a NetView subtask. This field may be defined by the subtask.
TIBAPWD	9	The VTAM password for NetView subtasks. This field may be defined by the subtask.
TIBAREA1	62	Pointers to other control blocks such as CWB, SWB, or PDB for NetView subtasks. This field may be defined by the subtask.
TIBCLRDF	1	Default color. X'00' Color field not found. X'F4' Green X'FA' Orange
TIBCLR#	1	Number of colors supported. 0 = default, 4 = base color, and 7 = extended color.
TIBEDATD	256	A scratch area for subtask use.
TIBECBPO	1	A one-byte constant to test whether an ECB has been posted.
TIBELT	4	A pointer to the subtask ECB list.
TIBEXLST	4	A pointer to the VTAM EXLST for NetView subtasks. This field may be defined by the subtask.
TIBFLGS	1	A byte of flags. TIBLRCNP = 1 means the RESUME routine has been newly promoted and should put out a new screen.
TIBHLITE	1	Highlight support flags. TIBBLINK = 1 means blinking is supported. TIBREVRS = 1 means reverse video is supported. TIBUNSCR = 1 means underscore is supported.
TIBMSGNM	8	The operator identifier of the subtask that issued the START TASK command. If the subtask was started automatically with INIT=Y, this field contains zeros.
TIBMUXIT	1	A counter used to track the level of multiple asynchronous interrupts.
TIBNDATD	256	A scratch area for subtask use.
TIBOSEXT	4	A pointer to an optional subtask extension to the TIB. The optional subtask releases any storage pointed to by this area.
TIBOSLST	4	The command processor LIST uses this field to display the status of an optional subtask.
TIBSAVEE	72	A save area for subtask use.
TIBSAVES	72	A save area for subtask use.

Field Name	Length	Description
TIBSCRID	4	Screen identifier. TIBSCRM 1-byte screen modification type. TIBSCRSN 3-byte screen serialization number.
TIBTVB	4	A pointer to the TVB. The address of the MVT can be obtained from the TVB. MVT locates all other control blocks.
TIBUFLD	4	This field is not referenced or changed by NetView. It may be defined by the subtask.
TIBXECB	4	NetView uses this field as an ECB for cross-domain communication in OST and NNT. It may be defined by the subtask.

TVB — Task Vector Block

The task vector block (TVB) contains information about the status of the subtask. There is one TVB for each subtask in NetView. Certain service routines, such as DSIPSS, use the TVB to store control information that is important for processing their code.

The TVB contains pointers to the MVT and the TIB. You can obtain the addresses of other important control blocks from these control blocks. The TIB, an extension of the TVB, represents an active task. TVBs are chained together through the TVBNEXT field. The beginning of the chain is pointed to by MVTTVB.

The relevant fields in the TVB are as follows:

Field Name	Length	Description
TVBCBH	4	A standard control block header. The CBHTYPE byte indicates the subtask type: X'00' PPT X'01' NNT X'02' OST X'03' HCT X'05' Optional subtask. To distinguish between different types of optional subtasks, refer to the field TVBMODNM.
TVBMECB	4	An event control block (ECB) that notifies the subtask that a message or a queue of messages has been sent using macro DSIMQS.
TVBMECBH	4	An event control block (ECB) that notifies the subtask that a high-priority message or queue of messages has been queued to the high-priority public message queue (TVBMPUBH).
TVBMECBL	4	An event control block (ECB) that notifies the subtask that a low-priority message or queue of messages has been queued to the low-priority public message queue (TVBMPUBL).
TVBMPRIQ	4	The address of the normal-priority private message queue.
TVBMPRQH	4	The address of the high-priority private message queue.
TVBMPRQL	4	The address of the low-priority private message queue.
TVBMPUBH	4	The address of the high-priority public message queue. See "Intertask Communication" on page 82 for information on processing public and private message queues.
TVBMPUBL	4	The address of the low-priority public message queue. See "Intertask Communication" on page 82 for information on processing public and private message queues.
TVBMPUBQ	4	The address of the normal-priority public message queue. See "Intertask Communication" on page 82 for information on processing public and private message queues.
TVBMVT	4	A pointer to the MVT.
TVBNEXT	4	A pointer to the next TVB on the TVB chain. The TVB chain is anchored from MVTTVB.

Field Name	Length	Description
TVBPRIQ	1	Priority queuing flags. TVBMM = 1 — The subtask provides services for the high-priority and low-priority message queues as well as the normal queue.
TVBRESTE	4	An event control block (ECB) that notifies the subtask that RESET command has been entered.
TVBTCB	4	The MVS task control block (TCB) address.
TVBTECB	4	An event control block (ECB) that requests that the subtask shut down as soon as possible. Include TVBECB in every subtask ECB list. A subtask may use this ECB to cause itself to shut down.
TVBTIB	4	A pointer to the TIB for the subtask.

The subtask uses the following bit fields. Some of these flag bits are defined by the subtask; others are defined by the main task.

Field Name	Length	Description
TVBAIIFR	4	Address of message buffer structure when command is driven from the message automation table. Otherwise, 0.
TVBHCUSE	4	May be defined by the subtask. For an HCT, this field tracks the number of subtasks currently using the hard-copy subtask.
TVBIND1	1	Indicator byte. TVBTERM — A value of 1 indicates that the subtask has ended normally, and the subtask has released all resources. This bit must be supported by the subtask. If the bit is set on by the main task before attaching the subtask, a value of 1 indicates to the subtask that it has been attached for cleanup. The subtask is to release all resources and return control to the main task with this bit still set to 1.
TVBIND2	1	Indicator byte. TVBVCLOS — This flag bit may be defined by the subtask. TVBABLOG — A 1 indicates a task is being re-initialized just after an ABEND.
TVBIND3	1	Indicator byte. TVBACTV = 1 — The subtask is active. This bit is set by the subtask. While this bit is on, messages may be sent to the subtask using macro DSIMQS. TVBINXIT = 1 — An IRB exit routine is running. TVBLGOFF = 1 — The subtask is ending upon request. TVBLGON = 1 — The subtask is starting. TVBRCVAI — May be defined by the subtask. For an OST or an NNT, TVBRCVAI = 1 means a RECEIVE ANY for cross-domain sessions has been issued. TVBRESET = 1 — Regular commands should stop processing immediately. If your subtask does not run command processors, you may redefine this flag.
TVBIND4	1	Indicator byte. TVBRCVRY — 1 means recovery is in progress for this subtask.

Field Name	Length	Description
TVBLUNAM	8	The value specified in the TSKID parameter of the TASK definition statement. This field is initialized before the subtask is attached.
TVBMEMNM	8	This field is initialized with the MEM parameter of the TASK definition statement of DSIDMN. It may be the name of the member of the data set DSIPARM (for MVS) or a file with filetype NCCFLST (for VM) that contains the initialization parameters for an optional subtask.
TVBMODNM	8	The name of the module to be attached as a subtask as specified in the MOD parameter of the TASK definition statement. This field may be used to determine the type of optional subtask.
TVBOPID	8	The unique subtask identifier. This name may be the same as TVBLUNAM. It is set up by the subtask when initialization is complete.
TVBUFLD	4	A user field that may be defined by the subtask.
TVBZIND2	1	Indicator byte. TVBABEND — The ABEND reinstate routine is running. If this indicator is on, macros DSIPUSH and DSIPOP cannot be issued. TVBLOGOF — The LOGOFF routine is running. If this indicator is on, DSIPUSH and DSIPOP cannot be issued. TVBRESUM — The RESUME routine is running.
TVBZIND4	1	Indicator byte. TVBAUTOO — User code is running under an unattended operator task. TVBAUTVE — The task depends on VTAM and must be terminated by the main task when VTAM ends. TVBAUTVS — The task depends on VTAM and must be reattached by the main task when VTAM starts. When VTAM ends, the task terminates.

USE — User Exit Parameter List

The user exit parameter list (USE) contains addresses for the following:

- The buffer containing the message
- The LU name associated with the message
- The operator identification
- The service work block (SWB)
- The task vector block (TVB)
- The parse descriptor block (PDB).

An extension of USE is present for DSIEX12 and the DST exit routines involved with input and output (XITCO, XITCI, XITVN, XITVI, XITVO, XITXL). For DSIEX12, the password, hard-copy printer name, and profile name are given. For the DST exit routines, the address of DSRB is given.

The relevant fields in the USE are as follows:

Field Name	Length	Description
USERCBH	4	A standard control block header. The second byte, USERCODE, indicates the exit routine that is called. Values X'01' – X'10' correspond to user exits DSIEX01 – DSIEX16. The DST user exit codes are defined in the USE control block.
USERLGON	36	An extension for DSIEX12 and the DST exit routines. If present, this extension contains the following fields:

USEDSRB	4	For DST exit routines, the fields XITVI, XITVO, XITCI, XITCO, and XITXL point to the DSRB associated with the DST I/O request. For other exit routines, this field is not initialized.
USENPSWD	8	For DSIEX12 only, this field contains the new password successfully entered by the operator at logon when OPTIONS VERIFY=MAXIMUM. If OPTIONS VERIFY=MINIMAL or OPTIONS VERIFY=NORMAL is specified, USENPSWD contains blanks (X'40'). For exit routines other than DSIEX12, this field is not initialized.
USERHCPY	8	For DSIEX12 only, this field contains the name of the hard-copy printer used by the operator for this session. If no hard copy is used or if OPTIONS VERIFY=MINIMAL is specified, USERHCPY contains blanks (X'40'). For exit routines other than DSIEX12, this field is not initialized.
USERPROF	8	For DSIEX12 only, this field contains the name of the profile used for this session. If OPTIONS VERIFY=MINIMAL is specified, USERPROF contains blanks (X'40'). For exit routines other than DSIEX12, this field is not initialized.
USERPSWD	8	For DSIEX12 only, this field contains the operator password entered at logon. If OPTIONS VERIFY=MINIMAL is specified, USERSWB contains blanks (X'40'). For exit routines other than DSIEX12, this field is not initialized.

Field Name	Length	Description
USERLU	4	A pointer to an 8-byte area that contains the logical unit name (LUNAME) related to the subtask in control, as follows: **Task** — **Name** MNT — The characters 'SYSOP'. OST — The node name of the operator's terminal. NNT — The APPL name of the OST that issued the START DOMAIN command (NCCFID DOMAINID followed by a 3-digit number). PPT — The NCCFID DOMAINID parameter followed by the characters 'PPT'. HCT — The node name of the hard copy-printer. DST — The name from the TSKID parameter of the TASK definition statement.
USERMSG	4	A pointer to a buffer in standard buffer format, consisting of a buffer header (BUFHDR) followed by text. For input-type exits, device dependencies have been removed. In exit routines DSIEX14, XITDI for end-of-file, and XITVN, this field is set to 0. In DSIEX04, the buffer is in the format set up by the caller. It has not yet been reformatted for the network log, MVS system log, or hard-copy log.
USEROPID	4	A pointer to an 8-byte area that contains a name related to the subtask in control, as follows: **Task** — **Name** **MNT** — The characters 'SYSOP'. **OST** — The operator identifier. **NNT** — The operator identifier. **PPT** — The NCCFID DOMAINID parameter followed by the characters 'PPT'. **HCT** — The node name of the hard copy-printer.
USERPDB	4	A field that points to a PDB or contains 0. The PDB contains parse data that relates to the buffer to which USERMSG points. For exit routines DSIEX02, DSIEX04, DSIEX07, DSIEX09, DSIEX10, DSIEX14, XITXL, and XITDI for end-of-file, this field contains 0. A PDB is not available when calling these exit routines.
USERSWB	4	A pointer to the SWB, which the exit routine uses as a work area or uses to request services from NetView. If necessary, another SWB may be obtained using macro DSILCS.
USERTVB	4	A pointer to the TVB. The TVB contains information about the subtask under which the exit routine was called. The TVB obtains the addresses of the TIB, the MVT, and the SVL (through the MVT).

Chapter 8

Chapter 8. Macro Reference

This chapter describes the purpose and coding of the NetView program macros. You may use these macros to request various service facilities when writing your own user exit routines, command processors, and subtasks. You must be in problem program state and user protection key for all these macros. NetView macros overwrite registers 0, 1, 14, and 15.

Prior to issuing any macro except DSICBS, you must set register 13 to a standard 72-byte save area.

All return codes in this chapter are shown in decimal.

Use only the parameters documented in this book. Any undocumented parameters that a macro may have are only for internal use by the NetView program and must not be used in user-written programs. Macro expansions are not part of the intended interface.

Notational Conventions

The following notational conventions apply to the macros described in this chapter.

UPPERCASE item
Uppercase letters represent parameters that you must code as shown.

lowercase **item**
Lowercase letters represent parameters for which you must supply the value, address, or name, rather than the literal information.

underscored item
An underscored item represents the default value of a particular parameter. If you specify no parameter, NetView uses the default value.

Braces { }
Small braces enclose the different options for a parameter. Large braces enclose mutually exclusive parameters; you must select one, and only one, of these parameters. Do not include the braces when coding the information.

Brackets [ ]
Brackets enclose an optional parameter. Optional parameters can be included or omitted independently of other parameters. Do not include the brackets when coding the information.

OR-sign |
The OR-sign separates the options for a required (brace-enclosed) parameter or for an optional (bracket-enclosed) parameter. For a required parameter, one of the options must be coded. For an optional parameter, none of the options have to be coded. Do not type the OR-sign when coding the information.

DSICBS – Control Block Services

Macro DSICBS includes DSECTs for the control blocks required for user-written programs during assembly.

DSICBS ensures that a control block is included only once, inner control blocks are included if necessary, and each definition for an inner control block precedes the definition of the outer control block. DSICBS also controls the format and printing, or suppression, of DSECTs for the control blocks.

MVT addressability is not required.

```
[name] DSICBS  [cbname,...]
               [,EJECT={YES|NO}]
               [,DEFER={ALL|THESE|INCLUDE|NO}]
               [,PRINT={YES|NO}]
```

cbname
Name of a control block, starting with DSI, to be included. Names must be separated by commas.

EJECT
Specifies that EJECT statements are performed between each control block expansion and after the last expansion.

DEFER
Defers control block expansions.

ALL
Specifies that all subsequent DSICBSs are not expanded until a DSICBS DEFER=INCLUDE is encountered. (If you specify this parameter, be sure to code DSICBS DEFER=INCLUDE later in the program.)

THESE
Specifies that these control block expansions are delayed until a DSICBS DEFER=INCLUDE is encountered.

INCLUDE
Specifies that any deferred control block expansions are to be expanded at this point in the program.

NO
Specifies that the control blocks are to be expanded immediately.

PRINT
Specifies that control blocks will be printed (expanded) in the assembler listing.

DSICES – Command Entry Services

Macro DSICES uses a specified buffer, parse descriptor block (PDB), or load module name to locate a command processor address.

DSICES locates a system command entry (SCE) that corresponds to the command verb and returns the SCE's address to a user-provided fullword area.

MVT addressability is required.

```
[name] DSICES  SWB={(register)|symbolic name}
              ⎧,BFR={(register)|symbolic name}|⎫
              ⎨,PDB={(register)|symbolic name}|⎬
              ⎩,MODNAME=modulename             ⎭
               ,SCTADDR={(register)|symbolic name}
               [,CLISTCK={YES|NO}]
```

SWB
Register, or symbolic name of a fullword area, containing the address of a service work block (SWB). SWBTIB identifies your TIB, which identifies your task type.

Note: BFR, PDB, and MODNAME are mutually exclusive, and one of the three must be specified.

BFR
Register, or symbolic name of a fullword area, containing the address of the buffer that contains the verb to be analyzed. This buffer must have an initialized BUFHDR.

PDB
Register, or symbolic name of a fullword area, containing the address of a completed parse descriptor block (PDB) to be used as input.

MODNAME
Specifies the module name to be located in the system command table. The *modulename* may be specified as the field that contains the module name or as the module name enclosed in single quotes.

Note: The first entry that is found in the SCT corresponding to the module name will be returned. If the module name specified is defined in DSICMD on more than one CMDMDL statement, then an unexpected address may be returned. In this situation, the BFR operand should be used instead.

SCTADDR
Register containing the address of a user-provided fullword area, or the symbolic name of that area, where the address of the system command entry corresponding to the module name or verb is to be returned. The returned pointer addresses an SCT entry (SCE) DSECT.

CLISTCK
Specifies whether to check for a valid command list name if the command has no CMDMDL statement in DSICMD.

Note: CLISTCK cannot be specified with MODNAME.

Return Codes in Register 15

0	Function was successful. One of the following occurred: 1. A regular command was found in the system command table and the address of the SCT entry was returned. 2. The verb was not found in the SCT (if CLISTCK was specified, a command list was found with the specified name), and the dummy SCT entry for a command list was returned.
4	The command found can be processed as a regular or immediate command; the address was returned.
8	An immediate command was found in the system command table; the address was returned.
12	The module was not found, or there was an invalid verb length; no address was returned.
16	Scope class failure. Either the operator was not authorized to issue the command, or storage could not be obtained to check the scope class. No address was returned.
20	Either the command found was incompatible with the task type that called the routine, and the address is returned, or CLISTCK=YES was specified and the request was issued in an asynchronous exit, and the address is not returned.
24	CLISTCK=YES was specified, but the command or command list was not found in DSISCT or DSICLD.
28	CLISTCK=YES was specified, but storage requested for CLISTCK processing could not be obtained.

DSIDATIM – Date and Time

Macro DSIDATIM obtains and formats the time and date.

DSIDATIM places the time and date in an output area. This macro can be used to obtain the time for the HDRTSTMP field of a message.

MVT addressability is required.

```
[name]  DSIDATIM   AREA={(register)|symbolic name}
                   [,FORMAT={EBCDIC|BINARY}]
```

AREA
Register containing the address of the area into which the date and time are returned, or symbolic name of that area. This area does not have a buffer header.

FORMAT
Specifies the format of the output.

EBCDIC
Returns the date and time in 17 bytes, formatted as follows:

```
mm/dd/yy hh:mm:ss
```

mm is the month, *dd* is the day, *yy* is the year, *hh* is the hours *mm* is the minute, and *ss* is the second.

BINARY
Returns the date and time in 8 bytes in packed decimal format as follows:

```
X'00yydddFhhmmss0C'
```

yy is the year and *ddd* is the Julian date. *hh* is hours, *mm* is minutes, and *ss* is seconds. The inital X'00', the middle X'F', and the ending X'0C' are characters which indicate the data is packed decimal.

Usage note: When using the BINARY form of DSIDATIM to initialize a HDRTSTMP, use an 8 byte work area for the AREA, and move the low order 4 bytes to HDRTSTMP.

DSIDEL — Delete User-Defined Module

Macro DSIDEL deletes user-defined load modules. You specify the name of the module to be deleted.

MVT addressability is not required.

```
[name]  DSIDEL  {EP=modulename                        }
                {EPLOC={(register)|symbolic name}     }
                [,DPR={(register)|MVTDPRAD(,MVTPTR)}]
```

EP
Specifies the name of the module to be deleted.

EPLOC
Specifies the address of an 8-byte field that contains the module name to be deleted. The module name should be left-justified and padded with blanks.

DPR (required for NetView Release 2 VSE only)
In VM/SP and MVS, this parameter is ignored. However, if you want to write a routine to compile and run on both a NetView Release 2 VSE system and on a NetView Release 3 MVS or VM/SP system, you will need to include this parameter. In VSE, this parameter specifies a register containing the address of the NetView dispatcher (DSIDPRNV), or MVTDPRAD(,MVTPTR), where MVTPTR is the symbolic name of a fullword area that contains the address of the MVT.

Return Codes in Register 15

Zero	Module has been deleted.
Nonzero	Attempt to delete module was unsuccessful. For more information, refer to *OS/VS Supervisor Services and Macro Instructions* or *CMS Commands and Macro Reference*.

DSIDKS – Disk Services

Macro DSIDKS can be used to connect to a DDNAME, locate a member, and read the records in that member. This macro can only be used to connect to data sets with the following DDNAMES: DSICLD, DSIPARM, DSIPRF, DSIVTAM, DSIMSG, CNMPNL1, BNJPNL1, BNJPNL2, or BNJMISC. (NetView opens these data sets, and keeps them open as long as it is up.) You can then use DSIDKS to find and read any member in any of the data sets concatenated under any of these DDNAMES in your NetView startup procedure.

You must have a copy of the DSECT for the disk service block (DSB) included in your program. MVT addressability is required.

```
[name]  DSIDKS   SWB={(register)|symbolic name}
                 ,DSBWORD={(register)|symbolic name}
                ┌,TYPE={CONN|FIND|DISC} ,NAME={(register)|┐
                {                          symbolic name}  }
                └,TYPE=READ                                ┘
```

SWB
Register, or symbolic name of a fullword area, containing the address of a service work block (SWB).

DSBWORD
Register containing the address of a user-provided fullword area on a fullword boundary, or symbolic name of that fullword area. When the macro completes processing for TYPE=CONN, this area contains the address of the disk service block (DSB). For other disk service requests, this area must specify the DSB address previously obtained by TYPE=CONN.

TYPE
Specifies the type of processing the service routine is to perform.

CONN
Specifies that the service routine is to connect to or access the caller's definition name (DDNAME). The address of the disk service block (DSB) is returned in the area specified by the DSBWORD parameter. DSIDKS must be issued with this option before any other options may be chosen.

FIND
Specifies that the service routine is to find the member specified by the NAME parameter. If the member is found, the first record is read. DSBBUFF addresses the buffer containing this record. Do not specify this option unless the CONN option has been specified.

DISC
Specifies that the service routine is to disconnect from the DDNAME and release the DSB.

READ
Specifies that the service routine is to read the next sequential record in the member. Do not specify this option unless the FIND option has been specified.

NAME

For TYPE=CONN and TYPE=DISC, a register containing the address of an eight-character user area with the caller's definition name (DDNAME), or the symbolic name of that area. The area should be left-justified and padded with blanks.

For TYPE=FIND, NAME is a register containing the address of an eight-character user area that contains the name of the member to be read, or the symbolic name of that area.

Return Codes in Register 15

For TYPE=CONN:

0	Function was successful. Data control blocks and I/O buffer were gotten and initialized.
4	Invalid data set name (for MVS) or filetype (for VM) was specified.
12	No storage was available for I/O buffer.

For TYPE=FIND:

0	Function was successful. Member or file was found and the first record was read.
4	Member or file was not found in the source statement library or in the specified library, or an empty member or file was found.
8	Member or file was found but an I/O error occurred on the first read.
12	Specified definition name or data set has not been opened.
20	Specified control block identifier was invalid; the member or file was not found.

For TYPE=DISC:

0	Disconnect was successful; data and I/O buffer were freed successfully.
4	Invalid filetype was specified. (VM only)
8	Disconnect failed. (VM only)
20	Specified control block identifier was invalid and no storage was freed.

For TYPE=READ:

0	Function was successful; record was read.
4	End of data was reached.
8	I/O error occurred during reading.
12	Reading this record is prohibited; an I/O error may have occurred, end of data may have been reached, or the caller did not issue TYPE=FIND first.
20	Specified control block identifier was invalid; the record could not be read.

DSIFIND – Find Long Running Command Storage

Macro DSIFIND retrieves a pointer to storage for a long running command processor and returns the storage address in register 1. A prior DSIPUSH instruction named the storage pointer.

If two storage pointers were given identical names, DSIFIND retrieves the most recent storage address with the specified name. DSIFIND can be issued in an immediate command. Refer also to macros DSIPUSH and DSIPOP.

MVT addressability is required.

```
[name]  DSIFIND   SWB={(register)|symbolic name}
                  [,LIST={(register)|symbolic name}]
```

SWB
Register, or symbolic name of a fullword area, containing the address of a service work block (SWB). The TIB address in SWBTIB must be correctly set.

LIST
Register containing the address of the parameter list used by the service routine, or symbolic name of that list. Do not specify this as register 1; register 1 contains the SWB address within DSIFIND. Do not put this list in the SWB that is to be passed to DSIFIND.

The parameter list is mapped by SWBLRCPL and contains the following fields.

Hex Offset	Length	Field
0	4	SWBLRCLN (Length)
4	16	SWBLRCNM (Name)

Where:

SWBLRCLN
Specifies the parameter list length. Set SWBLRCLN equal to SWBLRCFI (decimal 20).

SWBLRCNM
Specifies the name of the storage to be located. The storage address is to be returned to register 1. Specify this field exactly as you specified it in the corresponding macro DSIPUSH. Instruction for specifying the name field are under "DSIPUSH – Establish Long Running Command" on page 202.

Return Codes in Register 15

0 Function was successful; storage pointer was retrieved and storage address was returned.

32 Invalid macro invocation. Fix assembly errors before trying to run the program.

36 Specified NAME was not found.

DSIFRE – Free Storage

Macro DSIFRE must be used to release storage that was obtained using macro DSIGET. Storage that was not obtained with DSIGET cannot be released using DSIFRE. Optionally, DSIFRE dequeues the storage from the user's task vector block (TVB).

Registers 2 through 12 may be used for register notation. DSIFRE always generates reentrant code.

MVT addressability is required.

```
[name]  DSIFRE   LV={n|(register)}
                 ,A={(register)|symbolic name}
                 [,SP={(register)|number}]
                 [,Q={YES|NO}]
                 [,TASKA={(register)|symbolic name}]
                 [,AQ={YES|NO}]
```

LV
Number of bytes, or a register that contains the number of bytes, of storage to be freed. This option is ignored if Q=YES is specified.

A
Register, or symbolic name of a fullword area on a fullword boundary, containing the address of the storage to be freed.

SP
Subpool number, or a register loaded with the subpool number, from which the storage is to be freed. Values 0 through 255 are acceptable; 0 is the default value.

Q
Indicates whether the storage is to be dequeued from the user's TVB. The option specified for this parameter must correspond to the option specified for the Q parameter in DSIGET that was used to obtain storage.

Note: Do not modify the two words immediately preceding the queued storage. Otherwise, an ABEND may result.

TASKA
Register containing the address of the task vector block (TVB), or symbolic name of the TVB for this task. If this parameter is not specified, the default is DSITVB, and addressability to the DSITVB is required.

AQ
Indicates whether all queued storage is to be released. If you specify AQ you cannot specify LV or A.

Return Codes in Register 15

The MVS and VM/SP return codes for DSIFRE are in Register 15:

0	Function was successful; storage was freed and dequeued, if requested.
4	Storage was found on the queue and was dequeued but was not freed (FREEMAIN failure).
20	Storage was not found on the queue.

DSIGET – Get Storage

DSIGET obtains storage. Optionally, DSIGET can be used to queue the obtained storage to your task vector block (TVB). Storage obtained with DSIGET must be released using DSIFRE.

Registers 2 through 12 may be used for register notation. DSIGET is intended to allow you to queue storage on the TVB chain so that NetView can free the storage at logoff.

MVT addressability is required.

```
[name] DSIGET  LV={n|(register)}
               ,A={(register)|symbolic name}
               [,SP={(register)|number}]
               [,LOC={RES|BELOW|ANY|TEST}]
               [,BNDRY={PAGE|DBLWD}]
               [,CLEAR={NO|YES}]
               [,Q={YES|NO}]
               [,TASKA={(register)|symbolic name}]
```

LV

Number of bytes, or register containing the number of bytes, of storage to be obtained. The value must be positive.

A

Register containing the address of the fullword area on a fullword boundary into which the address of the obtained storage is returned, or symbolic name of that fullword area.

SP

Subpool number, or register containing the subpool number, from which the storage is to be obtained. Values 0 through 255 are acceptable; 0 is the default value.

LOC

In VM and in MVS/370, this parameter is ignored. In MVS/XA, this parameter specifies where to allocate storage.

RES
Allocate storage that is consistent with the residency of the caller.

BELOW
Allocate storage below 16 Mb.

ANY
Allocate storage anywhere.

TEST
The caller of DSIGET has set the high order bit of register 15 to indicate the type of storage desired. A 0 in the high order bit means allocate storage below 16 Mb; a 1 means allocate storage anywhere.

BNDRY

Specifies the alignment of obtained storage.

PAGE
Specifies that obtained storage is to be aligned on a page boundary.

DBLWD
Specifies that obtained storage is to be aligned on a doubleword boundary.

CLEAR
Specifies whether the allocated storage is to be initialized to zero.

Q
Indicates whether the obtained storage is to be queued to your TVB. The option specified for this parameter must correspond to the option specified for the Q parameter in DSIFRE that is used to free the storage.

TASKA
Register containing the address of the TVB for this task, or symbolic name of that TVB. If this parameter is not specified, the default is DSITVB and addressability to the DSITVB is required.

Return Codes in Register 15

The MVS and VM/SP return codes for DSIGET are in Register 15.

0	Function was successful; storage was obtained.
4	No storage was obtained.

DSIKVS – Keyword/Value Services

Macro DSIKVS is used in a command processor to determine whether an operator is authorized to use a given keyword or value.

The return code shown in register 15 indicates whether the operator who issued the command has been authorized to issue it with the particular keyword, value, or both.

MVT addressability is required.

```
[name] DSIKVS  SWB={(register)|symbolic name}
              {,CMD={(register)|symbolic name}       }
              {,SCTADDR={(register)|symbolic name}   }
               ,KEYWORD={(register)|symbolic name}
               [,VALUE={(register)|symbolic name}]
```

SWB
Register, or symbolic name of a fullword area, containing the address of a service work block (SWB).

CMD
Register containing the address of an 8-byte field with the command name left-justified and padded with blanks, or symbolic name of that field.

SCTADDR
Register, or symbolic name of a fullword area, containing the address of the SCT entry for the command that is to be checked.

KEYWORD
Register containing the address of an 8-byte field, or the symbolic name of an 8-byte field, that contains the keyword, left-justified and padded with blanks. The parameter is required.

VALUE
Register containing the address of an 8-byte field, or the symbolic name of an 8-byte field, that contains the value, left-justified and padded with blanks. VALUE is specified when VALCLASS checking is desired.

Note: If both KEYWORD and VALUE are specified, KEYWORD is scope-checked before VALUE. If KEYWORD results in a nonzero return code, VALUE is not checked.

Return Codes in Register 15

0	The specified keyword and value, if given, are in the operator's scope of commands.
4	The specified keyword was not in this operator's scope of commands.
8	The specified value was not in this operator's scope of commands.
12	A required parameter was missing, or an invalid parameter was specified in DSIKVS.
16	Working storage could not be obtained. No storage is available.

DSILCS – Obtain/Release Control Blocks

Macro DSILCS performs one of the following actions:

- Obtains an SWB for the caller and places the address of that SWB in a fullword area specified by the CBADDR parameter
- Releases an SWB
- Obtains a command work block (CWB) for the caller and places the address of that CWB in a fullword area specified by the CBADDR parameter
- Releases a CWB
- Locates a TVB by operator identification or by LU name
- Locates, from a specified starting position, the next active TVB for a NetView-NetView task (NNT), a hard-copy task (HCT), an operator station task (OST), or an optional task
- Locates a TVB for an operator designated as a receiver of authorization messages by the AUTH statement of a profile definition.

MVT addressability is required.

```
[name] DSILCS  CBADDR={(register)|symbolic name}
               [,LOC={RES|ANY|TEST|BELOW}]
             ⎧ ,CWB={GET|FREE}                          ⎫
             ⎪ ,SWB={GET|FREE}                          ⎪
             ⎪ ,TVB={(register)|symbolic name}          ⎪
             ⎨ ⎧,LU={(register)|symbolic name}   ⎫      ⎬
             ⎪ ⎨,OPID={(register)|symbolic name} ⎬      ⎪
             ⎪ ⎩,NEXT={OST|HCT|NNT|OPT|PPT}      ⎭      ⎪
             ⎩ ,AUTHRCV={YES|NO}                        ⎭
```

CBADDR
For the GET and TVB options, register containing the address of a user-provided fullword area on a fullword boundary, or symbolic name of that area. The specified SWB, CWB, or TVB address is returned to this area.

For the FREE option, CBADDR contains the control block address.

LOC
Used with CWB=GET or SWB=GET to determine the residency of the work block you are requesting (for MVS/XA only).

RES
Obtain a work block in storage consistent with the residency of the caller.

ANY
Obtain a work block anywhere in storage.

BELOW
Obtain a work block below 16 Mb.

TEST
The caller of DSILCS has set the high order bit of register 15 to indicate the type of storage desired. A 0 in the high order bit means allocate storage below 16 Mb, and a 1 means allocate storage anywhere.

CWB

Specifies the type of operation to be performed on the CWB.

GET

Specifies that the caller needs a CWB. The address of the CWB is returned to the area specified by the CBADDR parameter. Before you request NetView services, you must initialize the CWBTIB field with the address of your TIB.

FREE

Specifies that the caller wishes to release the CWB whose address is found in the area specified by the CBADDR parameter.

SWB

Specifies the type of operation to be performed on the SWB.

GET

Specifies that the caller needs an SWB. The address of an SWB is returned to the area specified by the CBADDR parameter. Before you request NetView services, you must initialize the SWBTIB field with the address of your TIB.

FREE

Specifies that the caller wishes to release the SWB whose address is found in the area specified by the CBADDR parameter.

TVB

Value is either:

1. Register containing the address of the TVB where the routine begins the search for the TVB specified by LU, OPID, or NEXT
2. Symbolic name of an area containing the address of this TVB.

TVB **must** be used with LU, OPID, or NEXT, and TVB **must not** be used with SWB, CWB, or AUTHRCV=YES.

The address of the beginning of this TVB chain is found in the MVTTVB. The TVB address found is placed in the area specified by CBADDR after the routine has completed processing.

Note: The routine searches the address once to the end of the TVB chain; it does not loop to the beginning of the TVB chain.

LU

Used with TVB, register containing the address of an 8-byte LU name field, or symbolic name of that field. This name locates a TVB with a matching LU name.

OPID

Used with TVB, register containing the address of an 8-byte operator identification field, or symbolic name of that field. This name locates a TVB with a matching operator identification.

NEXT

Used with TVB, specifies the TVB to be located for the next task.

OST

Specifies that the TVB associated with the next active operator station task is to be located.

HCT

Specifies that the TVB associated with the next active hard-copy log task is to be located.

NNT
Specifies that the TVB associated with the next active cross-domain task is to be located.

OPT
Specifies that the TVB associated with the next optional task is to be located.

PPT
Specifies that the TVB associated with the next active primary programmed operator interface task is to be located.

AUTHRCV
Specifies that the routine is to search for the first TVB to locate an operator authorized to receive messages related to successful and unsuccessful logons and lost station messages. See the discussions of the AUTH statement and unsolicited message routing in *NetView Administration Reference*.

Return Codes in Register 15

0	Function was successful. The address was returned, or the control block was released.
4	No active TVBs of the type specified were found.
8	If TVB was specified, end of the TVB chain was reached, or invalid OPID was provided.
8	If SWB=GET or CWB=GET was specified, no storage was available.
8	If SWB=FREE or CWB=FREE was specified, defective control block.
12	Invalid parameters were passed to DSILCS.

DSILOD – Load User-Defined Module

Macro DSILOD loads a user-defined module. You specify the name of the module to be loaded.

MVT addressability is not required.

```
[name] DSILOD {EP=modulename|                          }
              {EPLOC={(register)|symbolic name}        }
              [,DCB={(register)|symbolic name}]
              [,LISTA={(register)|symbolic name}]
              [,DPR={(register)|MVTDPRAD(,MVTPTR)}]
```

Note: EP or EPLOC must be specified, but not both.

EP
Specifies the name of the module to be loaded.

EPLOC
Register, or symbolic name of an 8-byte field, containing the *modulename* to be loaded. The *modulename* should be left-justified and padded with blanks.

DCB
Register, or symbolic name of an area, containing the address of the DCB for a partitioned data set to be searched for the module. For MVS/XA, the DCB must reside below 16 Mb. The DCB is ignored in VM.

LISTA
Register containing the address of a 64-byte BLDL directory entry list, or symbolic name of that list. In VM/SP and MVS, this parameter is ignored. It is retained only for compatibility with NetView Release 2 VSE.

DPR (required for NetView Release 2 VSE only)
In VM/SP and MVS, this parameter is ignored. However, if you want to write a routine to compile and run on both a NetView Release 2 VSE system and on a NetView Release 3 MVS or VM/SP system, you will need to include this parameter. In VSE, this parameter specifies a register containing the address of the NetView dispatcher (DSIDPRNV), or MVTDPRAD(,MVTPTR), where MVTPTR is the symbolic name of a fullword area that contains the address of the MVT.

Return Codes in Register 15

Zero	Module has been loaded.
Nonzero	Module has not been loaded.

If the module is successfully loaded, register 0 contains the load point address of the module. Register 1 contains the authority code in the high-order byte and the module length (in double words for MVS and VM) in the low-order three bytes.

If the module has not been loaded, register 15 contains the return code returned by the system load facility. Register 1 contains the ABEND code and register 0 contains the reason code for the ABEND.

The module is loaded in virtual storage consistent with the linkage editor attribute RMODE. If AMODE=24, the module is called in 24-bit mode, otherwise it is called in 31-bit mode.

DSIMBS – Message Buffer Services

Macro DSIMBS puts variable text combined with message text into a buffer that you provide. DSIMBS can determine the size of the buffer required to accommodate the message to be built.

You can supply variable fields to be inserted into NetView messages or unique messages of your own with up to nine varying positional fields.

MVT addressability is required.

```
[name] DSIMBS  SWB={(register)|symbolic name}
              ⎧,MID={nnn|(register)|symbolic name|*equatename}⎫
              ⎪     ⎡,P1= (text,lngth[,padlng,side,fill])⎤    ⎪
              ⎪     ⎢ .                                  ⎥    ⎪
              ⎨     ⎢ .                                  ⎥    ⎬
              ⎪     ⎢ .                                  ⎥    ⎪
              ⎪     ⎣,P9= (text,lngth[,padlng,side,fill])⎦    ⎪
              ⎩,MSGA=(pdb1addr,pdb2addr)                      ⎭
              ⎧,BFR={(register)|symbolic name}     ⎫
              ⎩,MSGSIZE={(register)|symbolic name} ⎭
               [,MSGTBL={(register)|symbolic name}]
               [,OPT=CONCAT]
```

SWB
Register, or symbolic name of a fullword area, containing the address of a service work block (SWB).

MID
Identifies the message to be edited for the user. The message may be specified by the message number (*nnn*), in a register, in a user area specified by symbolic name, or by the equate name preceded by an asterisk. For example, for MSG999 EQU 999, you could specify MID=MSG999.

P1...P9
Used only in combination with the MID parameter, these values specify the positional fields in a message that are to be replaced by user-supplied text. The first two values, *text* and *lngth*, must be specified; the others are optional.

text
Address of the variable text or the symbolic name of the area with the text to be substituted into the edited message.

lngth
Length of the variable text to be substituted into the edited message. The maximum length is 255 characters, specified in character format. It can be a binary value in a register or in a user area specified by symbolic name. The user area must be a 4-byte fullword.

padlng
Length of the variable field to be padded with fill characters. This length must be equal to or less than the length specified by the *lngth* parameter. The maximum length is 255 characters, specified in character format. It can be a binary value in a register or in a user area specified by symbolic name. The user area must be a 4-byte fullword.

side

May be specified as L for left-fill or R for right-fill. The default is R.

fill

The fill character for the area to be padded. The default fill character is a blank (hex 40).

MSGA

The following registers are used for variable field substitution in message texts:

pdb1addr

Address of the PDB or the symbolic name of a fullword area with the address of the PDB. This address contains the addresses and lengths of the variable fields to be substituted into the message text.

Note: Because the variable field information is contained in *pdb1addr*, the P1...P9 parameters may not be used with MSGA.

pdb2addr

Address of the PDB or the symbolic name of a fullword area with the address of the PDB. This address contains the message skeleton to be edited. This is not a NetView message; you supply the message.

BFR

Register, or symbolic name of a fullword area, containing the address of the buffer in which the edited message is to be returned. BFR must have an initialized BUFHDR. Macro DSIMBS initializes the HDRMLENG, HDRDOMID, and HDRTSTMP fields.

MSGSIZE

Register containing the address of a user-provided fullword area, or symbolic name of that area. Use MSGSIZE only to request the service routine for determining the size of the buffer needed for the message to be edited. When the routine has completed processing, the required size is returned in this area.

MSGTBL

Register, or symbolic name of a fullword, containing the address of a user-defined message table. Macro DSIMDS generates this table.

OPT

Tells NetView to search DSIMDM for a specified message identifier that cannot be found in the specified private message table (MSGTBL). If OPT=CONCAT is omitted, NetView searches only the private message table for the message identifier.

Return Codes in Register 15

0 Function was successful. (1) The edited message is in the provided buffer and the length of the message is stored in the message length field of the buffer header, or (2) the size of the message buffer required has been calculated and stored in the area specified by MSGSIZE.

4 The edited message is in the provided buffer, but the message skeleton contained a parameter for which the caller did not supply text. The message contains the characters &*n* where *n* may be from 1 – 9.

8 Unsuccessful. The buffer overflowed, and the message has been truncated. The size of the truncated message has been stored in the message length field of the buffer header.

12	The message number specified could not be found in the NetView or user-specified message table. The message 000I was edited into the caller's buffer. If only the buffer size was requested, the size of message 000I is returned.
16	The caller did not supply a buffer address.
20	Combined conditions 4 and 8 occurred.
24	Combined conditions 8 and 12 occurred.
28	A validity check failed on the user message definition module. The address passed in the MSGTBL parameter does not point to a message definition module that was created with macro DSIMDS.
32	Storage request failed.
36	I/O error.
40	Unexpected end-of-file found.

DSIMDS — Message Definition Services

Macro DSIMDS generates a message definition module to be used by macro DSIMBS to display messages in user exits, command processors and subtasks.

After a message definition module has been coded, it must be assembled and link-edited into a NetView load library. DSIMDS has no return codes.

MVT addressability is not required.

Three forms of DSIMDS are required to generate a message definition module. These forms are described in the following three formats; they must be coded in the sequence shown.

Format 1: Start Message Definition Module Statement

```
name   DSIMDS   prefix
                ,TYPE=START
                ,SEARCH={T|D|B}
                [,MAXLEN={71|142|213}]
```

name
Required parameter that starts the message definition module. The name specified becomes the CSECT name for the module. The name can be any valid name which does not conflict with an existing NetView load module name.

prefix
Required positional parameter that becomes the 3 character prefix for the messages in the module. The prefix should not conflict with the NetView message prefixes (AAU, BNJ, CNM, DSI, or DWO).

TYPE
Specifies the beginning of generation for the message definition module.

SEARCH
Indicates where the messages can be found: in the message definition module, on disk, or both. SEARCH causes a message definition module to be built. This indicator becomes part of the message definition table.

T
Indicates the messages can be found only in the message definition module. (Default)

D
Indicates the messages can be found only on disk. Individual message statements using Format 2 of DSIMDS should not be coded for the messages. The individual messages are coded on disk. Refer to "Defining Messages on Disk" on page 183.

B
Indicates some of the messages can be found in the message definition module, and the others can be found on disk. Individual message statements using Format 2 of DSIMDS should only be coded for those messages that will not be defined on disk. Refer to "Defining Messages on Disk" on page 183.

MAXLEN
Defines the maximum message length. MAXLEN is determined by calculating the length of "pppxxxt msgtext" (message number, type, a blank, and text). MAXLEN should be a multiple of 71. Maximum message size allowed is 213 characters. If MAXLEN is not specified, message size defaults to 142 characters.

Format 2: Define Individual Messages Statement

```
[label]  DSIMDS   xxx,'message text [&&n]'
                  ,TYPE={A|I}
```

label
Optional label.

xxx
Message number. It may be any number from 000-999. When DSIMBS is issued to build the message, it will create the message identifier by concatenating your three-character prefix, the message number, and the type.

When coding your message CSECT, you must code a message 000 statement to be issued when an invalid message number is specified. Message 000 should have one insert (&&1) which will contain the invalid message number. You may want to use wording similar to NetView's message DSI000I:

```
MSG000 DSIMDS 000,'MESSAGE &&1 ISSUED BUT DOES NOT EXIST IN
MESSAGE TABLE DSIMDM - CALL IGNORED', TYPE=I
```

You should replace DSIMDM with the name of your message table definition module; that is, the name specified on the DSIMDS TYPE=START statement. The DSIMBS service routine will substitute the message number of the invalid message in place of &&1.

Note: At coding time, be sure that the buffer passed to DSIMBS is big enough to hold the message with all the inserts substituted. Otherwise, the message will be truncated.

message text
Text of the message added or changed.

&&n
Optional information may be substituted in the message at the place where the &&n occurs. The positional fields are specified by the Pn parameter in the DSIMBS macro. &&1 - &&9 may be specified.

TYPE
Specifies the message type.

A
Specifies an action message, one for which appropriate action must be taken.

I
Specifies that the message is for information only. No specific action is required.

Format 3: End Message Definition Statement

```
DSIMDS    TYPE=END
```

TYPE
Specifies the end of the message definition module. This is the last statement specified.

Note: The TYPE=END statement should be followed by an assembler END statement.

Defining Messages on Disk

Messages can be defined on disk instead of, or in addition to defining them in the message definition module. The benefits of defining messages on disk are:

- The message coding is simpler and does not require assembling and link-editing the message definition module when messages are added or changed.
- The messages are read in by NetView when the DSIMBS macro is issued. Messages can be changed while NetView is running and the changes take effect immediately.

  Note: Messages that will be issued from code running with TVBINXIT on cannot be read from disk. These messages must be defined in your message definition module with Format 2 DSIMDS statements.

Messages are coded in members in the NetView DSIMSG data set (filetype NCCFLST, for VM). The member names (file names, for VM) must be in the format of DSI*pppxx* where:

ppp is the message prefix which was defined on the DSIMDS start message definition module statement.

xx is the first two digits of the message number. Up to ten messages can be defined in a member (xx0-xx9).

The message syntax for the user messages is:

xxxt message text [&n]

xxx
Message number. It may be any number from 001-999.

t
Message type:

A
Specifies an action message, one for which appropriate action must be taken.

I
Specifies that the message is for information only. No specific action is required.

message text
Text of the message added or changed.

&n

Optional information may be substituted in the message at the place where the &n occurs. The positional fields are specified by the Pn parameter in the DSIMBS macro. &1 - &9 may be specified.

User Message Definition Module Example

Following is an example of defining five user messages (USR001 - USR005). Message USR001 will reside in the message definition module called USRTABLE. Messages USR002 - USR005 will reside in the NetView DSIMSG data set under the member name DSIUSR00.

Message Table Definition Module USRTABLE

```
USRTABLE DSIMDS USR,TYPE=START,SEARCH=B
MSG000   DSIMDS 000,'USER MESSAGE &&1 ISSUED BUT DOES EXIST IN MESS*
               AGE TABLE USRTABLE - CALL IGNORED.',TYPE=I
MSG001   DSIMDS 001,'THIS IS USER MESSAGE 1',TYPE=I
         DSIMDS TYPE=END
         END
```

NetView DSIMSG Member DSIUSR00

```
002I THIS IS USER MESSAGE 2
003A THIS IS USER MESSAGE 3, RETURN CODE = &1
004I THIS IS USER MESSAGE 4, &1 IS TODAY'S DATE
005I THIS IS USER MESSAGE 5, TIME IS &1
```

DSIMQS – Message Queuing Services

Macro DSIMQS sends a user-supplied message or command to the message queue of a task's TVB.

This message or command appears on the operator's screen or hard-copy log, depending upon which identification is specified. Buffers that are formatted as internal function requests (IFRs) are not displayed. Instead, they cause the receiving subtask to take the action requested by the IFR. Buffers that are formatted as IFRs should not be sent to the authorized receiver (see AUTHRCV operand below).

MVT addressability is required.

```
[name] DSIMQS  SWB={(register)|symbolic name}
              ⎧,BFR={(register)|symbolic name}          ⎫
              ⎪,TASKID={(register)|symbolic name|PPT}   ⎪
              ⎨,AUTHRCV={YES|NO}                        ⎬
              ⎪,LIST={(register)|symbolic name}         ⎪
              ⎩[,EXCEPT={(pointer)|symbolic name}]      ⎭
               [,BFRFLG={YES|NO}]
               [,PRI={NORMAL|HIGH|LOW|(register)|symbolic name}]
```

SWB
Register, or symbolic name of a fullword area, containing the address of a service work block (SWB).

BFR
Register, or symbolic name of a fullword area, containing the address of a buffer. (If DSIGET obtained this buffer, the DSIFRE must free it after use. Use the same option for the Q parameter for both DSIGET and DSIFRE.) BFR must have an initialized BUFHDR.

Note: Specify one, and only one, of the following: TASKID, AUTHRCV=YES, or LIST.

TASKID
Register containing the address of a user-provided 8-byte area, or symbolic name of that area, or PPT for the Primary POI task. The area should contain the 8-character operator identification (TVBOPID) of the task for which the message is to be queued.

AUTHRCV
Specifies that the first operator designated as the receiver of authorized messages (by the AUTH statement of profile definition) is to receive the message. All messages sent to the authorized receiver are routed first to the PPT to test for automation and message routing (via the ASSIGN command). If not suppressed by automation or handled by routing, the messages are sent to the authorized receiver, if one is logged on, or to the system console. The AUTHRCV option must only be used for messages. If the buffer to be sent is formatted as an internal function request (IFR) and it is not an automation IFR containing a message, the DSIMQS macro will fail with a return code 4 (invalid buffer format) in register 15. See the discussions of the AUTH statement and unsolicited message routing in *NetView Administration Reference*.

LIST

Pointer to, or symbolic name of, a fullword area containing the address of a list of operator IDs or group IDs to receive the message. Operators are assigned to groups using the ASSIGN command. See *NetView Operation* for more information on the ASSIGN command.

- If 1ST is specified as the LIST type, the first logged on operator in the list will receive the message. (The first logged on operator may be in a group.) The receiving operator ID is returned in the MQSENTTO field in the SWB.
- If ALL is specified as the LIST type, and a return code of 0 was received, the message was sent to all specified operators and groups of operators in the list who were logged on. While the message was sent to all specified logged on operators, it was not necessarily received.
- If multiple operator or group IDs are specified, the last two fields (shown in the ID list below) must be repeated for each operator or group listed.

The ID list has the following format:

Hex Offset	Contents
0	1ST or ALL (three bytes)
3	Number of IDs in list (one byte)
4	Unused (eight bytes)
C	Operator or group ID (eight bytes)

EXCEPT

May be specified only if LIST is specified. This parameter specifies an eight-character field that contains an operator or group ID, left justified and padded with blanks, that should not receive the message.

BFRFLG

Specifies whether the subtask that sends the buffer has released control and responsibility for it (BFRFLG=YES). (With BFRFLG=YES, the buffer must include HDRMCEXT with HDRSENDR initialized.) BFRFLG=NO indicates that the receiving subtask is to make a copy of the buffer and return it.

PRI

Specifies a priority for message processing by the destination task. The value is either one of the three strings HIGH, NORMAL, or LOW, or a register or name of a full word area containing one of the three values defined in DSISWB: MQSHI, MQSNORM, or MQSLO. The default value is NORMAL. A message or command sent at HIGH priority would begin processing after any normal message currently in progress, but before other queued NORMAL messages. A message or command sent at NORMAL priority would similarly pre-empt a queue of LOW priority messages. Of the NetView supplied tasks, only destination task types OST, PPT, and NNT respect priority. For destination tasks that have not indicated support for multiple priorities, DSIMQS automatically converts all messages to NORMAL priority.

Return Codes in Register 15

0	Function was successful; the message is queued.
4	The format of the buffer that was passed was invalid.
8	The task is inactive, or no task was identified as a receiver for the buffer.
12	A buffer could not be obtained.
16	NetView is terminating.
20	SWB address is invalid.
22	The list specified with the LIST option contained no operator IDs. It contained only unassigned group IDs.
23	Messages were routed to the first 255 operators and/or groups.
24	An invalid value was specified for priority.

DSIOIS – Operator Identification Services

Macro DSIOIS searches the operator identification table (DSIOIT) in preparation for span authority checking using DSISSS.

DSIOIS returns the relative position of the entry to a user-provided fullword area.

MVT addressability is required.

```
[name] DSIOIS   SWB= {(register)|symbolic name}
                ,OPID= {(register)|symbolic name}
                ,OITPOS= {(register)|symbolic name}
```

SWB
Register, or symbolic name of a fullword area, containing the address of a service work block (SWB).

OPID
Register containing the address of an 8-byte, left-justified operator identification field; or symbolic name of that field.

OITPOS
Register containing the address of a fullword area, or symbolic name of that area. When the routine has located the specified operator identification in DSIOIT, that entry's relative position is returned to this fullword area. For example, the third entry results in a fullword 3 being returned.

Return Codes in Register 15

0	Function was successful; the position of the entry is returned.
4	Unsuccessful. The entry was not found in DSIOIT.

DSIPAS – Parameter/Alias Services

Macro DSIPAS receives a command parameter as input and searches the system command table (SCT) to determine whether the entered parameter is an alias for the actual parameter.

If the parameter is an alias, the regular value is returned to a user-provided area. If it is not an alias, the input value is returned to the user area. If the value is invalid, blanks are returned to the input area.

MVT addressability is required.

```
[name]  DSIPAS   SWB={(register)|symbolic name}
                 ,PDB={(pdbreg)|pdbname,(entreg)|entname|'entry'}
                 ,OUT={(register)|symbolic name}
```

SWB
Register, or symbolic name of a fullword area, containing the address of a service work block (SWB).

PDB
Specifies two values. The first value is the address of a PDB and the second value is the entry number of the field in the PDB to be examined.

pdbreg
Register that contains the address of the PDB.

pdbname
Symbolic name of a fullword that contains the address of the PDB.

entreg
Entry number, right justified.

entname
Symbolic name of a fullword that contains the entry number, right justified.

entry
Constant that specifies the entry number.

Note: PDBCMDA must contain the address or pointer to an entry in the SCT.

OUT
Register containing the address of a user-provided 8-byte area to which the NetView equivalent of the input parameter is returned if found, or symbolic name of that user area.

Return Codes in Register 15

0	A regular parameter value was returned.
4	No equivalent was found; the same parameter is returned.
8	Invalid parameter; blanks are returned.

DSIPOP – Remove Long Running Command

Macro DSIPOP removes a long running command element that DSIPUSH placed on the stack.

The element canceled is the one nearest the top of the stack with the name specified in the parameter list. If a calling command procedure is suspended by DSIPUSH, use DSIPOP to let the command procedure continue at the next instruction. The command procedure continues when the RESUME routine or currently running command returns control to the OST or PPT.

Do not use DSIPOP while in a LOGOFF routine, an ABEND reinstate routine, or an immediate command.

Refer also to macros DSIFIND and DSIPUSH.

MVT addressability is required.

```
[name]  DSIPOP    SWB={(register)|symbolic name}
                  ,LIST={(register)|symbolic name}
                  [,COMPCDE={(register)|symbolic name}]
```

SWB
Register, or symbolic name of a fullword area, containing the address of a service work block (SWB). The TIB address in SWBTIB must be correctly set.

LIST
Register containing the address of a parameter list used by the service routine, or symbolic name of that list. Do not specify this as register 1; register 1 contains the SWB address within DSIPOP. Do not put this list in the SWB that is to be passed to DSIPOP.

The parameter list contains the following fields:

Hex Offset	Length	Field
0	4	SWBLRCLN (Length)
4	16	SWBLRCNM (Name)

Where:

SWBLRCLN
Specifies the parameter list length. Set SWBLRCLN equal to SWBLRCPO (decimal 20).

SWBLRCNM
Specifies the name of the storage to be dequeued and freed. Specify this field exactly as you specified it in the corresponding macro DSIPUSH. This 16-byte field is used as is. Instructions for specifying the name field are under "DSIPUSH – Establish Long Running Command" on page 202.

COMPCDE
Specifies the value of the completion code for the long running command (LRC) being removed. The value may be specified as a register, symbolic name of a full word area, or a full word literal. The value is meaningful only if the LRC

being removed specified a RESUME routine and only if the process that created the long running command element (using DSIPUSH) was directly invoked from another long running command. If you do not specify COMPCDE, the value defaults to:

0 if the LRC being removed is in control (top of stack) at the time DSIPOP is invoked. This is the usual (and recommended) case.

-5 if the LRC being removed is not at the top of the stack. Negative five is treated as a CANCEL request by NetView command procedures and certain related commands.

You can be certain that your LRC is at the top of the stack, **when it is resumed.**

Note: NetView command procedures are long running commands. If a long running command you write is called from a command procedure, you can pass a return code to it using the COMPCDE keyword. If a long running command you write makes a direct call to schedule a command procedure, you can obtain its return code upon resumption from CWBRCODE.

Return Codes in Register 15

0 Function was successful; the long running command processor element is dequeued.

16 Request issued while in an immediate command, or while NetView is currently in an exit, a LOGOFF, or an ABEND reinstate routine.

32 Invalid macro call. Fix assembly errors before trying to run the program.

36 Specified NAME was not found.

DSIPOS – ECB Post Services

Macro DSIPOS indicates the completion of an event by posting an event control block (ECB).

MVT addressability is required.

```
[name] DSIPOS     ecbaddress[,compcde]
```

ecbaddress
Symbolic name of ECB or register (1 – 12) that contains the address of the ECB. If a register is specified, it must be enclosed in parentheses.

compcde
Value of the completion code to be placed in the ECB (0 – 16,777,215) or in a register (0, 2 – 12) that contains the value. If a register is specified, it must be coded in parentheses. If no value is specified, 0 is assumed.

Note: These parameters are positional; they must be specified in the indicated order.

IPRS – Parsing Services

Macro DSIPRS parses commands using specified or assumed delimiters, or determines the size of the parse table required to parse the input buffer.

The parse table describes the data contained in the buffer. DSIPRS finds delimiters in the data and formats the PDB to indicate data segments separated by the delimiters. DSIPRS may be called again to build the parse table in a user-provided area.

MVT addressability is required.

```
[name]  DSIPRS    SWB={(register)|symbolic name}
                  ,BFR={(register)|symbolic name}
                 {,PDBSIZE={(register)|symbolic name}}
                 {,PDB={(register)|symbolic name}    }
                  [,FIRST={YES|NO}]
                  [,DELIM=('D1','D2',...'Dn')]
                  [,SUB={YES|NO}]
```

SWB
Register, or symbolic name of a fullword area, containing the address of a service work block (SWB).

BFR
Register, or symbolic name of a fullword area, containing the address of the buffer to be used for input. BFR must have an initialized BUFHDR.

PDBSIZE
Register containing the address of a fullword area to which the size of the parse table is to be returned, or symbolic name of that area.

PDB
Register containing the address of a fullword pointing to the area where the parse table is to be built, or symbolic name of that area. The parse table must include a user-initialized DSICBH header that contains the control block identification and length before the data can be parsed.

FIRST
Indicates whether the first word of the input buffer can be delimited only by a blank (YES) or by any delimiter (NO).

DELIM
Allows you to specify delimiters instead of NetView defaults. NetView default delimiters are blank, comma, period, and equal sign. Blank is always considered a delimiter, even if you specify your own delimiters.

SUB
Indicates whether all text within single quotes is to be parsed as one element (YES) or not (NO). This option treats everything between single quotes as one element provided the first quote is preceded by a delimiter and the last quote is followed by either a blank or comma.

For each example below, DSIPRS is issued to pass the given character string with the default delimiters and SUB=YES.

Example 1

RETURN CODE IS 'NON-ZERO'.

DSIPRS will return the UNBALANCED QUOTES return code.

Example 2

RETURN CODE IS 'GOOD', CONTINUE.

The following PDB table is built:

PDBTYPE	PDBLENG	PDBDISP	TOKEN VALUE
ƀ	6	24	RETURN
ƀ	4	2B	CODE
ƀ	2	30	IS
,	4	34	GOOD
.	8	3B	CONTINUE

Example 3

RETURN CODE IS (X'00'), CONTINUE.

The following PDB table is built:

PDBTYPE	PDBLENG	PDBDISP	TOKEN VALUE
ƀ	6	24	RETURN
ƀ	4	2B	CODE
ƀ	2	30	IS
'	2	33	(X
'	2	36	00
,	1	39	)
.	8	3C	CONTINUE

Return Codes in Register 15

0 Function was successful. The required size of the table was returned in PDBSIZE, or the command was parsed and the parse table was built.

4 The input buffer was parsed, but there was no data in the input buffer (zero length data) or the data in the input buffer was all blanks. Only the buffer address and number of entries (0) could be returned in the parse table.

8 The parse table size was too small for the input buffer; a partial parse table was built, and the number of entries was set to the number that the parse table could hold. The size of the parse table should be increased.

12 Unbalanced quotes. Returned only if SUB=YES is specified.

16 The number of characters between two consecutive delimiters in the input buffer was greater than 255.

20 An unpaired Kanji delimiter or uneven number of Kanji data bytes was found in the input buffer. For example, one of the following may have occurred:

- The end of the input buffer was found before the Kanji data-entering delimiter (SI).
- A second Kanji data-beginning delimiter (SO) was found before the Kanji data-ending delimiter (SI).
- An uneven number of Kanji data bytes was found between Kanji data delimiters.

100 No PDB or an invalid PDB was passed.

DSIPSS — Presentation Services

Macro DSIPSS writes a message to an operator's screen or sends messages to another NetView. The presentation service routines, which DSIPSS calls, control screen formats, organize data into a specific form for each device, and send the data.

For MVS console tasks, system WTO services are used to display messages in addition to processing as described for unattended operator tasks.

MVT addressability is required.

```
[name] DSIPSS  SWB={(register)|symbolic name}
               [,APPLID={(register)|symbolic name}]
               ,TYPE={OUTPUT|FLASH|IMMED|XSEND|
                  SCRSIZE|WINDOW|ASYPANEL|
                  PANEL|CANCEL|PSSWAIT|
                  TESTWAIT}
               [,ECBLIST= {(register)|symbolic name}]
               [,BFR= {(register)|symbolic name}]
               [,SIZE= {(register)|symbolic name}]
               [,PANEL= {(register)|symbolic name}]
               [,OPTIONS={MSG|SEG|
                  FIRST|MIDDLE|
                  LAST|ONLY}]
```

SWB
Register, or symbolic name of a fullword area, containing the address of a service work block (SWB).

APPLID
Register containing the address of an 8-byte area that contains the name (left justified and padded with blanks) of the application program to which the data is to be sent, or symbolic name of that 8-byte area. This name should be the same as the name specified on the START command when a session is started. APPLID is specified only when TYPE=XSEND is specified.

TYPE
Type of presentation services routine to be called:

OUTPUT routine
Specifies that the routine is to send a message to the operator's terminal. Do not use this option in immediate command processors, in user exit routines with TVBINXIT set on, or in user exit DSIEX12. The maximum message length before truncation is 32,767 characters for OST and 256 characters for NNT. Upon completion of the macro, the length of the text in the HDRMLENG field of the BUFHDR is set to the length of the data after any trailing blanks have been truncated.

DSIPSS calls DSIEX02A, message copy (ASSIGN), and logging functions.

FLASH
These messages are not suppressed by &WAIT processing, or message automation, nor are they logged (they can be logged prior to calling DSIPSS if you choose). They will be displayed regardless of the STIFLE state.

Note: For OUTPUT and FLASH, if your data is formated as an automation IFR, it must be in buffers obtained with DSIGET using Q=NO and subpool zero. DSIPSS will free the automation IFR structure. If your data is NOT formatted as an automation IFR, there is no restriction on the type of storage used and the storage will not be freed. You are responsible for freeing it.

IMMED
Specifies that the routine is to send a message to the operator station's immediate message area. The maximum message length before truncation occurs is 71 characters. Use this option only in immediate command processors or the DSIEX01 user exit routine. When this parameter is specified, no message header information is sent to the display screen. TYPE=IMMED terminates full-screen mode and causes subsequent terminal input to be treated as commands.

DSIPSS calls DSIEX02A, message copy (ASSIGN), and logging functions.

XSEND
Specifies the routine is to send data to another NetView with which a session exists. (Sessions are started with the START DOMAIN = command.) The maximum data length before truncation is 240 characters.

SCRSIZE
Specifies the routine is to return the screen size in row-column format.

WINDOW
Requests information on the size of the output area of the standard screen. This option is valid only from an OST. Under any other task, the request is considered null; register 15 contains a return code of 0, but no function is performed. Three output area sizes are returned:

- Minimum
- Current
- Maximum.

The minimum window size may be used to produce screens that are independent of the current window size. The current window shows the screen size currently in effect. The maximum window size is useful for calculating the maximum storage needed to produce title-line panels.

ASYPANEL
Specifies that the issuing routine assumes control of the screen. Input and output will be formatted as 3270 data stream commands. Notification of input availability is done asychronously by the posting of ECBs.

After a DSIPSS TYPE=ASYPANEL, input to the terminal will be treated as input to the process issuing the ASYPANEL request until either a DSIPSS TYPE=CANCEL or a DSIPSS TYPE=OUTPUT is issued.

Full-screen mode is not supported for unattended operator tasks, including those associated with a system console.

PANEL
Although this option is supported for compatibility with earlier releases of NetView, it is not recommended.

CANCEL
Cancels pending asynchronous full-screen input. This option is used when changing the characteristics of the asynchronous full-screen processor, such as the ECB address or the panel address. TYPE=CANCEL is allowed only from an OST. This option can be invoked regardless of whether a DSIPSS

TYPE=ASYPANEL is active or the input from TYPE=ASYPANEL has been posted as complete.

After TYPE=CANCEL is issued, no further input is received from the terminal until TYPE=OUTPUT, TYPE=IMMED, TYPE=PANEL, or TYPE=ASYPANEL is issued.

PSSWAIT
Specifies that a command is to wait for a list of its own events and a list of events that should be allowed to interrupt the command events. TYPE=PSSWAIT is allowed only from an OST.

Note: Use macro DSIWAT if you do not want the command to wait for the completion of events.

TESTWAIT
Allows a command processor to test whether an event has occurred that should interrupt the asynchronous full-screen command processor. TYPE=TESTWAIT is allowed only from an OST. This option can be used before a DSIPSS TYPE=ASYPANEL is issued to determine if the asynchronous full-screen panel input/output should be attempted. If DSIPSS TYPE=PSSWAIT is used to wait for events, this option can prevent unnecessary screen input/output by allowing testing before panel input/output is requested.

ECBLIST
For TYPE=PSSWAIT, register, or symbolic name of a fullword area, containing the address of an ECB list. An ECB list is a list of addresses of user-defined event control blocks that is copied and combined with an ECB list. NetView waits for this combined list; when one of the events associated with this list is posted, control is returned to the next sequential instruction. The input ECB list is made up of fullword ECB addresses. The last address in the list must have the first bit set on to specify that this is the last entry.

BFR
Register, or symbolic name of a fullword area, containing the address of a user-provided buffer. This buffer should contain the data to be processed. BFR is used only for TYPE=FLASH, TYPE=OUTPUT, TYPE=IMMED, and TYPE=XSEND. BFR must have an initialized BUFHDR.

SIZE
For TYPE=SCRSIZE, register, or symbolic name of a fullword area, containing the address of a user-provided 4-byte area to contain the size of the display screen, in row-column format. For example, a 1920-character screen is defined as X'00180050', since the screen is 24 rows (X'0018') by 80 characters (X'0050').

For TYPE=WINDOW, a register containing the address of a 12-byte area, or symbolic name of that area. The window size is returned in binary to the area. The window size is the number of lines available for output on the screen. The size varies depending on screen size and the number of input lines specified on the INPUT command. The format of the area is:

Bytes (Decimal)	Bytes (Hex)	
0	(0)	
2	(2)	Minimum Window Size, Rows
4	(4)	Minimum Window Size, Columns
6	(6)	Current Window Size, Rows
8	(8)	Current Window Size, Columns
10	(A)	Maximum Window Size, Rows
12	(C)	Maximum Window Size, Columns

PANEL

For TYPE=ASYPANEL, a register containing the address of a 20-byte parameter list, or the symbolic name of that list. The parameter list is formatted as follows:

Bytes (Decimal)	Bytes (Hex)		
0	(0)	ECB Adress	
4	(4)	Output Data Stream Address	
8	(8)	User Input Area Address	
12	(C)	Output Length	Input Area Length
16	(10)	Data length Address	
20	(14)		

If asynchronous full-screen output is requested, the output data stream address field contains the address of a 3270 data stream including a 3270 command, WCC, and orders to be written to the terminal. The command must be coded using remote EBCDIC values. The output length field indicates the length, in bytes, of the 3270 data stream (32,767 bytes maximum). If output is requested, the ECB address, input area length, user input area address, and data length address fields are not used.

To read asynchronous full-screen input from a terminal, the ECB address area contains the address of an event control block to be posted when the asynchronous input is received. The user input area address contains the address of a user area into which the full-screen panel data is read. If the length of the data being read is greater than the user input area, the data will be truncated in that area. The input area field indicates the length of the input data area in bytes (32,767 bytes maximum). The data length address field contains the address of a halfword field set to the amount of data read when the ECB is posted.

OPTIONS

Specifies the type of message to be sent. OPTIONS is used only for TYPE=OUTPUT. The default is MSG, which specifies that the data to be sent is a complete message.

In general, title line output should be used instead of the other values for the OPTIONS parameter. The other values are as follows:

- SEG, which specifies that the data to be sent has no message header.
- FIRST, which begins full-line mode. It specifies that the data is to start at the top of the screen, with full 80-byte line width.
- MIDDLE, which continues full-line mode.

- LAST, which specifies the end of a full-line mode screen. The screen is locked until the operator signals for the screen to be refreshed.
- ONLY, which specifies that one full-line message is to be written at the top of the screen with the rest of the screen blank.

Return Codes in Register 15

0 Function was successful; the message is written. For TYPE=PSSWAIT, an ECB has been posted. Check the ECB list to determine which event has completed. For TYPE=ASYPANEL, the send or receive request has passed NetView syntax and buffer checking and has been sent to VTAM; it does not indicate the success or failure of VTAM completion of the receive. The ECB post code must be checked to determine the success or failure of the ASYPANEL request. The post code will be put into the ECB specified in the panel parameter list.

4 For TYPE=XSEND, no RPL was found and no data was sent.

8 Parameter error. There is an error in the formatting of the message buffer header. For TYPE=XSEND, the session is not active and no data is sent. For TYPE=PANEL, the input or output length is invalid, that is, greater than 32,767 bytes (X'7FFF'). For TYPE=ASYPANEL, the parameter list is inconsistent. If the output buffer is specified, its length must also be specified. If the input ECB is specified, the input area address, input area length, and the data length address of the returned length must be specified.

12 There is not enough storage available in NetView to complete the request. No output is sent, and the input command processor is not be scheduled.

16 DSIPSS TYPE=OUTPUT was issued for an immediate command or in an IRB exit routine. Use DSIPSS TYPE=IMMED or DSIMQS instead. Too many OPTIONS=MIDDLE were specified, and the screen is full. This OPTIONS=MIDDLE is treated as an OPTIONS=LAST. If another MIDDLE is issued, it is treated as an OPTIONS=FIRST. The screen wraps around, and return code 24 is issued. The requested DSIPSS service could not be performed under this unattended operator task, MVS console operator task, PPT, or DST.

20 No terminal session exists. For TYPE=PANEL, the panel request came from a task other than an OST. No output is sent, and the input command processor is not scheduled. For TYPE=ASYPANEL, the panel request came from a task other than an OST. No input will be received. For TYPE=CANCEL, the panel request came from a task other than an OST.

24 For OPTIONS=FIRST, MIDDLE, LAST, or ONLY, options were specified in an incorrect order.

28 For OPTIONS=FIRST, MIDDLE, LAST, or ONLY, user exit DSIEX02 specified that title-line output was to be deleted. The output was not written to the screen. This return code does not indicate a severe error, but rather warns that protocol for title-line mode may have been violated.

32 For TYPE=PANEL, no input command processor is scheduled. The operator requested escape to NetView mode by selecting option 1 when prompted by message DSI817A.

36 For TYPE=PANEL or TYPE=ASYPANEL, a temporary error occurred. The contents of the screen have been modified. Reformat the screen using an Erase/Write or Erase/Write Alternate 3270 command. Then retry the request.

40	A permanent input/output error occurred. Do not retry the request. No output will be sent, and no input processor will be scheduled. For TYPE=ASYPANEL, no input will be received. For TYPE=CANCEL, NetView is unable to restart normal terminal activity.
44	For TYPE=PANEL, no input is scheduled, because the operator requested reset by selecting option 3 when prompted by message DSI817A.
48	For TYPE=ASYPANEL, no input/output is scheduled because the command processor issued a second DSIPSS TYPE=ASYPANEL requesting input before the previous request had completed.
56	For TYPE=PSSWAIT or TYPE=TESTWAIT, at least one NetView ECB was posted.

The ECB post codes for PSS TYPE=ASYPANEL are found in the event control block if one was specified. They are as follows:

0	Function was successful; the requested data is available.
12	There is not enough storage available in NetView to complete the request. The output data was sent, but the input data is not available.
36	A temporary error occurred during a full-screen read. Retry the request. The output data was sent, but the input data is not available.
40	A permanent error occurred during a full-screen read. Do not retry the request. The output data was sent, but the input data is not available.
52	The requested input was canceled by DSIPSS TYPE=CANCEL. Do not retry the request immediately. The output data was sent, but the input data is not available.

DSIPUSH – Establish Long Running Command

Macro DSIPUSH can perform either of two related functions:

- Establish *named storage*

  A storage pointer passed to DSIPUSH is associated with a 16 character name chosen by the DSIPUSH caller. After a successful DSIPUSH, the DSIFIND macro can be used to obtain the storage pointer from the name. All task types supporting commands also support named storage (PPT, OST, NNT, DST).

- Establish *resumable command*

  A resumable command can return to its caller, with the assurance that the specified RESUME routine will be called when other scheduled activity for the task allows. Resumable commands may exit, for example, to wait for data requests to be satisfied or to allow queued "simulated terminal" commands to execute. All NetView components behave this way. Task types supporting regular commands also support resumable commands (PPT, OST, NNT).

Both of these functions are "task level" operations, global for the task where they occur, but invisible from all other tasks. All requests define recovery[4] and termination procedures.

MVT addressability is required.

When a command issues DSIPUSH for a RESUME routine, the following applies:

If a command invoked by a command procedure (NetView command list language, REXX, or high-level language) specifies MAJOR when issuing DSIPUSH, the command procedure is suspended at that point until the RESUME routine is removed by DSIPOP. If such a command specifies MINOR when invoking DSIPUSH, then the RESUME routine will be invoked following the completion of the command procedure.

A command may schedule a command procedure **and obtain a return code** by taking the following steps:

1. invoke DSIPUSH for its own RESUME routine,
2. make a **direct call** to schedule the command procedure,
3. upon return from the direct call, exit to allow the command procedure to complete
4. when the RESUME routine gains control, check the return code from the command procedure in CWBRCODE.

Notes:

1. When a command procedure is canceled for any reason, its return code is -5.
2. WAIT and PAUSE statements in a command procedure called by a long running command do not cause premature calling of the RESUME routine. The RESUME routine is not scheduled until the command procedure completes.

Do not use RESUME or ABEND reinstate routines under a DST.

[4] Note that DST tasks always terminate after any failure (ABEND), thus recovery routines are not appropriate under DST's.

Do not use DSIPUSH while running in a LOGOFF or ABEND reinstate routine specified on a previous DSIPUSH, or while TVBINXIT is on.

```
[name] DSIPUSH SWB={(register)|symbolic name}
               ,LIST={(register)|symbolic name}
               [,ROLL={YES|NO}]
               [,PROMOTE={YES|NO}]
```

SWB

Register, or symbolic name of a fullword area, containing the address of a service work block (SWB). The TIB address in SWBTIB must be correctly set.

LIST

Register containing the address of a parameter list used by the service routine, or symbolic name of that list. Do not specify this as register 1; register 1 contains the SWB address within DSIPUSH. Do not put this list in the SWB that is to be passed to DSIPUSH. DSIPUSH will read, but not write to, your parameter list; nevertheless, reentrancy demands you put the parameter list in dynamic storage if your code updates any part of it (such as the storage pointer) at run time.

The parameter list contains the following fields:

Hex Offset	Length	Field
0	4	SWBLRCLN = Length
4	16	SWBLRCNM = Name
14	4	SWBLRCST = Storage address
18	8	SWBLRCRE = RESUME routine
20	8	SWBLRCAB = ABEND reinstate routine
28	8	SWBLRCLG = LOGOFF routine
30	4	SWBLRCFG = Flags

Where:

SWBLRCLN

Specifies the parameter list length. Set SWBLRCLN equal to SWBLRCPU (decimal 52).

SWBLRCNM

Associates a name with your Long Running Command or storage address. You will use this name on subsequent calls to DSIFIND or DSIPOP. The name should be unique within a particular task, but a duplicate name used under another task does not interfere. A second use of DSIPUSH for named storage with the same name under the same task will temporarily hide the first storage pointer. The first storage pointer becomes accessible (via DSIFIND) after DSIPOP is issued for the duplicate name.

The name can be any combination of bits as long as you specify the name identically for all macro calls with the same name. Names beginning with DSI are reserved for NetView names. The name field is used as is; it is not padded or justified.

SWBLRCST
Can be used to associate a storage address with the specified name. However, NetView makes no use whatsoever of the value specified; it is only returned when DSIFIND is invoked, specifying the same name.

SWBLRCRE
Specifies the load module name of the RESUME routine. If this field is 0, no RESUME routine is indicated. The load module named must have a corresponding CMDMDL statement.

SWBLRCAB
Specifies the load module name of the ABEND reinstate routine. This name must not be 0 for tasks other than DST tasks. This name must be 0 for DST tasks. The load module named must have a corresponding CMDMDL statement.

SWBLRCLG
Specifies the load module name of the LOGOFF routine. This name must not be 0. The load module named must have a corresponding CMDMDL statement.

SWBLRCFG
Indicates whether the DSIPUSH execution is minor or major. Bit 0 is the indicator bit; it is defined when a RESUME routine is specified. When bit 0 = 1, a minor DSIPUSH is perfomed. When bit 0 = 0, a major DSIPUSH is perfomed. See "RESUME Routines" on page 64 for more information on major and minor invocations. When no RESUME routine is specified, this field is ignored.

ROLL
Specifies whether a long running command processor element has strict, global dependencies on other LRCS.

All those LRCS created with ROLL=NO will be serviced in strict FIFO order for RESUME, ABEND reinstate, and LOGOFF. LRCS created with ROLL=YES are regarded as being interrelated if they are all created during a single command invocation or if they are created during a RESUME routine call. Note that if a command processor or RESUME routine makes a direct call (see "Calling a Command Directly" on page 21) to another command processor, DSIPUSH still regards that command's processing as part of the original command's processing. All LRCS related in this way are processed in FIFO order, with respect to each other. The processing of other, unrelated LRCS, may be in any order.

When no RESUME routine is specified, you must specify ROLL=NO or allow the value to default. When a RESUME routine *is* specified, ROLL=YES is the default and is strongly recommended.

PROMOTE
When YES is specified, a search is made of all previous DSIPUSHS (excepting those since canceled via DSIPOP). If a DSIPUSH with the same name is found, then the entire, related group associated with that name will be made the active group.

If the group does not exist, the request is treated as an ordinary DSIPUSH and a stack element is created.

The storage specified in the invocation becomes the current storage associated with the specified name (SWBLRCNM) and previous storage associated with the name will be returned to the caller in register 0.

PROMOTE may only be used with ROLL=YES (or by default) and only when a RESUME routine is specifed.

When PROMOTE=NO is specified and when the PROMOTE parameter is omitted, no search is performed. The DSIPUSH is considered *new* and an LRC (possibly duplicate) is created.

Return Codes in Register 15

0 Function was successful; the long running command request is queued.

4 Storage is not available for request.

8 ABEND reinstate or LOGOFF routine required but not specified.

12 Request issued from invalid task:

- RESUME request issued under DST
- ABEND request issued under DST
- DSIPUSH issued and task is not OST, NNT, DST, or PPT.

16 Request issued while in an immediate command or while NetView is currently in an exit, or in the middle of a LOGOFF routine or ABEND reinstate routine.

20 RESUME routine is a command list or the CMDMDL statement did not pass validity checking.

24 ABEND reinstate routine is a command list or the CMDMDL statement did not pass validity checking.

28 LOGOFF routine is a command list or the CMDMDL statement did not pass validity checking.

32 Invalid macro invocation. Fix assembly errors before trying to run the program.

Note: If register 15 contains return code 20, 24, or 28, register 0 will contain a secondary return code. See "DSICES – Command Entry Services" on page 163 for an explanation of the return code in register 0.

DSIRDS – Resource Definition Services

DSIRDS locates the specified resource in the authorization and routing table (ART) and returns the address of the ART entry to a user-provided fullword area.

MVT addressability is required.

```
[name] DSIRDS  SWB={(register)|symbolic name}
               ,LUNAME={(register)|symbolic name}
               ,ARTPOS={(register)|symbolic name}
               [,STATUS={ACT|INACT}]
```

SWB
Register, or symbolic name of a fullword area, containing the address of a service work block (SWB).

LUNAME
Register containing the address of a user-provided area, or symbolic name of that area. The area should contain the 8-byte, left justified LUNAME to be located in the ART.

ARTPOS
Register containing the address of a fullword area, or symbolic name of that area. When the routine has located the specified entry in ART, that entry's address in the table is returned to this area.

STATUS
Specifies whether the LUNAME entry in ART is to be marked as active (ACT) or inactive (INACT).

Return Codes in Register 15

0	Function was successful; the entry was found and its address was returned.
16	No ART.
20	Unsuccessful. The specified entry was not found in ART, or the entry is inactive.

DSIRXCOM — Access REXX Variables

An assembler language program called from a REXX command list may wish to access the variables within the REXX command list which invoked it. Macro DSIRXCOM provides an interface to the GCS EXECCOMM macro to obtain access to these variables. DSIRXCOM issues a GCSCALL SETCOMM macro to insure that the correct REXX Work Block is accessed and then issues the GCSCALL EXECCOMM macro to access and manipulate the variables.

Before invoking DSIRXCOM, you must build a chain of shared variable request blocks (SHVBLOCK) as described in the *VM/SP System Product Interpreter Reference*.

```
[name] DSIRXCOM  TIB={(register)|symbolic name}
                 ,SHVB={(register)|symbolic name}
```

TIB
Register containing the address of the task information block (TIB) for this task or symbolic name of the TIB.

SHVB
Register containg the address of the first shared variable request block (SHVBLOCK) or symbolic name of the SHVBLOCK.

Return Codes in Register 15

0	Successful.
-1	Invalid entry conditions.
-2	Insufficient storage.
-3	No EXECCOMM entry point found.

Note: For a more detailed explanation of the non-zero return codes, see the discussion of EXECCOMM in the *VM/SP System Product Interpreter Reference*.

DSIRXEBS — Get An EVALBLOK

Macro DSIRXEBS is an interface to check the required interface parameters for getting a VM evaluation block (EVALBLOK).

The macro DSIRXEBS can be used instead of the TSO/E macro IRXRLT for the TSO/E user. This macro is to be used by VM/REXX users for getting evaluation blocks.

The size of the EVALBLOK data area is passed. The header size is added to the data size, the appropriate doubleword area is obtained, and initialization of the area is performed. Parameters are passed in the system work block (SWB) which you provide. If you pass an EVALBLOK address in the EVBPTR, that block is freed before the new block is obtained.

MVT addressability is required.

For additional information on DSIRXEBS, see Chapter 6 on page 109.

```
[name]  DSIRXEBS   SWB={(register)|symbolic name}
                   ,LENGTH={(register)|symbolic name}
                   ,EVBPTR={(register)|symbolic name}
                   ,ENVBPTR={(register)|symbolic name}
```

SWB
Register, or symbolic name of a fullword area, containing the address of an SWB.

LENGTH
Register, or symbolic name of a fullword area, containing the address where the EVALBLOK data size is obtained.

EVBPTR
Register, or symbolic name of a fullword area, containing the address where the EVALBLOK address is placed when it is obtained. If this area is not zero (X'00') initially, it is assumed that it contains an EVALBLOK address to be freed.

ENVBPTR (MVS/XA only)
Register, or symbolic name of a fullword area, containing the address of the TSO/E environment block address.

Return Codes in Register 15

0	Function was successful; the address of the EVALBLOK was returned in EVBPTR. If an EVALBLOK address was passed, the block was freed.
8	Storage was insufficient to obtain the EVALBLOK. If an address of an EVALBLOK was passed, the storage was freed.
12	An invalid request was made. One of the required parameters was not correctly specified.
20	Processing was not successful. A new evaluation block was not allocated (MVS/XA only).
28	Processing was not successful. A valid language processor environment could not be located for the current task (MVS/XA only).

DSISSS – Search Span Name Table Services

Macro DSISSS determines whether an operator has authority to control a particular resource.

DSISSS checks a specified bit position in the span name table (SNT) and returns the address of the first entry whose corresponding bit is set to 1. (See Figure 17.) The address is returned to a user-provided fullword area.

MVT addressability is required.

```
[name]  DSISSS   SWB={(register)|symbolic name}
                 ,OITPOS={(register)|symbolic name}
                 ,SNTADDR={(register)|symbolic name}
```

SWB

Register, or symbolic name of a fullword area, containing the address of a service work block (SWB).

OITPOS

Register containing the address of a user-provided fullword area, or symbolic name of that area. This area should contain the bit position to be checked for the first bit set to 1 in the SNT.

Note: The bit positions in the SNT correspond to entry positions in the operator identification table (OIT); for example, the first bit corresponds to the first entry in the OIT.

SNTADDR

Register containing the address of a user-provided fullword area, or symbolic name of that area. On input, this area should contain the address of the entry in the SNT where the search is to begin. When the routine has completed processing, this area contains the address of the first entry that the search encountered whose corresponding operator bit was set to 1. The starting address specified in SNTADDR may also be stored elsewhere in case relative location calculations are necessary for searching the authorization and routing table (ART).

The span name table (SNT) is illustrated below.

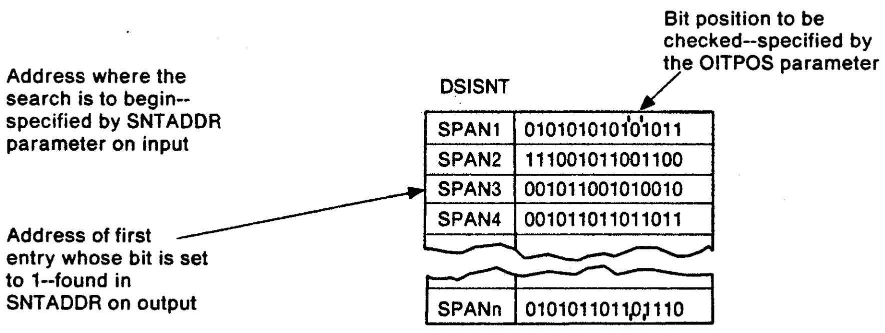

Address where the search is to begin--specified by SNTADDR parameter on input
Address of first entry whose bit is set to 1--found in SNTADDR on output
Bit position to be checked--specified by the OITPOS parameter
DSISNT
SPAN1 010101010101011
SPAN2 111001011001100
SPAN3 001011001010010
SPAN4 001011011011011
SPANn 010101101101110

Figure 17. Span Name Table (SNT)

Return Codes in Register 15

0	Function was successful; an entry was found and its address was returned.
12	Unsuccessful. No entry was found. The address originally submitted is still in the area specified by SNTADDR.

DSISYS – Operating System Indicator

Macro DSISYS is used for testing the current operating system. It is meant for use in programs that are to be run on multiple operating systems, to vary compilation according to the current system. It allows for system dependent code to be placed in common programs.

```
DSISYS
```

&DSISYST

Global variable that must be declared. &DSISYST is set at compilation time by the DSISYS macro to reflect the current operating system. Use the AIF assembler statement to test its value. Possible values are:

Value	Operating System
VS2/MVS	MVS/370
MVS/XA	MVS/XA
VM/SI	VM/370
DOS	VSE

DSIWAT — ECB Wait Services

Macro DSIWAT causes a subtask to wait for completion of an event.

MVT addressability is not required.

```
[name]  DSIWAT   {ECB={(register)|symbolic name}    }
                 {ECBLIST={(register)|symbolic name}}
                  [,DPR={(register)|MVTDPRAD(,MVTPTR)}]
```

ECB
Register (2 — 12) containing the address of an aligned fullword to be used as an event control block (ECB), or symbolic name of that aligned fullword.

ECBLIST
Register containing the address of a contiguous list of fullword addresses of ECBs, or symbolic name of that list. The last entry in the list of ECB addresses has the high-order bit 0 set to 1 to indicate the end of the list.

DPR (required for NetView Release 2 VSE only)
In VM/SP and MVS, this parameter is ignored. However, if you want to write a routine to compile and run on both a NetView Release 2 VSE system and on a NetView Release 3 MVS or VM/SP system, you will need to include this parameter. In VSE, this parameter specifies a register containing the address of the NetView dispatcher (DSIDPRNV), or MVTDPRAD(,MVTPTR), where MVTPTR is the symbolic name of a fullword area that contains the address of the MVT.

The following example shows how DSIWAT can be coded:

```
          DSIWAT ECBLIST=LISTAREA
           :
ECB1      DC F'0'
ECB2      DC F'0'
LISTAREA  DC A(ECB1)
          DC A(ECB2)
          DC A(ECB3)
          DC A(ECB4+X'80000000')
           :
ECB3      DC F'0'
ECB4      DC F'0'
```

Execution resumes when any one ECB is posted. The DSIPOS macro is used to set bit 1 of the ECB to 1. A completion code can also be set in the low-order three bytes of the ECB.

DSIWCS – Write Console Services

Macro DSIWCS writes a message to the system operator's console. However, this macro should not be used to output a DBCS message. DBCS messages are not operating system supported.

The message will be truncated at 120 characters. The message buffer must have an initialized buffer header.

MVT addressability is required.

```
[name]  DSIWCS     SWB={(register)|symbolic name}
                   ,BFR={(register)|symbolic name}
```

SWB
Register, or symbolic name of a fullword area, containing the address of a service work block (SWB).

BFR
Register, or symbolic name of a fullword area, containing the address of a buffer with the message. This buffer must have an initialized BUFHDR.

Return Code in Register 15

0	Function was successful; the message was sent to the system console.

DSIWLS – Write Log Services

Macro DSIWLS writes a record to the network log, the hard-copy log, the MVS system log, an external log, or a NetView sequential log depending on the use of the DEFAULTS command, the OVERRIDE command, DSIEX02A, or DSIEX04 as follows:

- Without the EXTLOG parameter, DSIWLS can send records to the network log, the MVS system log, the operator's hard copy device or with the SAMREC parameter, to a NetView sequential log. For the network log, the record may be truncated, depending on the user-defined VSAM record size. For the MVS system log, system truncation rules apply.
- With the EXTLOG parameter, DSIWLS sends records to an external task that can log the data.

MVT addressability is required.

```
[name] DSIWLS  SWB={(register)|symbolic name}
              ⎧,BFR={(register)|symbolic name}                              ⎫
              ⎪       ⎡⎧,HCT={(register)|symbolic name}                ⎫⎤   ⎪
              ⎨       ⎣⎩,EXTLOG= 'xxx'|(register)|symbolic name}       ⎭⎦   ⎬
              ⎪,SAMREC={(register)|symbolic name}                           ⎪
              ⎪          ,SAMLEN={(register)|symbolic name}                 ⎪
              ⎩          ,SAMTASK={(register)|symbolic name}                ⎭
```

SWB
Register, or symbolic name of a fullword area, containing the address of a service work block (SWB).

BFR
Register, or symbolic name of a fullword area, containing the address of a user-provided input buffer. This buffer should contain the record that is to be logged. This buffer must have an initialized BUFHDR.

Note: You must specify either BFR or SAMREC, but not both.

HCT
Register, or symbolic name of a fullword area, containing the hard copy task's TVB address.

Note: You may specify either HCT or EXTLOG with BFR, but not both.

EXTLOG
Indicates that the buffer is to be logged externally under the DSIELTSK subtask. See *NetView Installation and Administration Guide.*

External logging can be accomplished in one of the following ways:

- Write to an SMF data set. This option is restricted to MVS systems. The data to be logged must be a standard System Management Facilities (SMF) record, with an SMF record type greater than or equal to 128. The DST XITXL user exit is called prior to writing the record to SMF.
- Write to a data set other than SMF. This option is not restricted to any particular operating system. The user must code and install the DST XITXL user exit. This user exit then performs the actual logging.

Note: You may specify either EXTLOG or HCT with BFR, but not both.

xxx
Three characters that become the last three characters of the command name used to select the command processor that will log the data. The first characters of the verb must begin with DSIEL. For example, if EXTLOG='ABC', the CMDMDL statement in DSICMD must be:

```
DSIELABC CMDMDL MOD=DSIELSMF,TYPE=D
```

Register
Register that contains the address of a 3-byte area that represents the last three letters in the name of an external logging command processor.

Symbolic name
Symbolic name of a 3-byte area that represents the last three letters in the name of an external logging command processor.

SAMREC
A keyword that either points to or names the record that should be written to the NetView sequential log which is controlled by the task indicated by the SAMTASK keyword. Unlike the BFR keyword, this parameter should only point to the data that is to be logged. NetView services will put the record into the correct format for scheduling the record to the sequential log task.

Note: You must specify either BFR or SAMREC, but not both.

SAMLEN
Register, or symbolic name of a fullword area, containing the length of the record to be logged. If the value of SAMLEN is greater than 32,000, the record to be logged will be truncated and the macro will return a non-zero return code.

The BLKSIZE for a sequential logging task must be at least as large as SAMLEN plus 36 or the record to be logged will be truncated. SAMLEN is required with SAMREC.

SAMTASK
Register, or symbolic name of a 2 fullword area, containing the name of the NetView task that has been defined to do sequential logging. This task will be verified for sequential logging capability. SAMTASK is required with SAMREC.

Return Codes in Register 15

If EXTLOG and SAMREC are **not** *specified*:

0	Function was successful; the record has been sent to the network log and to the hard-copy log.
4	No storage is available for logging.
8	Successful log to the hard-copy log; network log not active.
12	Successful log to the network log; hard-copy log not active for this task.
16	The hard-copy log for this task and this network log are both inactive.

If SAMREC is specified:

0	Function was successful; the record has been queued to the sequential log task.
4	No storage is available.

16 The task is not active, no data set is available, or the specified task is not a sequential log task.

If EXTLOG *is specified*:

0 Function was successful. A copy of the caller's buffer has been queued to the external logging task (DSIELTSK).

4 No storage is available for copying the user input buffer for logging.

24 No external logging command processor was found.

28 DSIMQS failed attempting to send the log record to the external logging task.

DSIZCSMS – CNM Data Services

Macro DSIZCSMS is a data services macro. It requests and sends CNM data over the CNM interface.

DSIZCSMS embeds the caller's network services request/response unit (RU) in a Forward RU that is passed to the SSCP over the access method's CNM interface. The SSCP then sends the embedded RU to the specified destination. For more information about SNA's RUs, see *Systems Network Architecture Formats.*

MVT addressability is not required.

```
[name] DSIZCSMS   SWB={(register)|symbolic name}
                  ,DSRB={(register)|symbolic name}
                  ,INPUT={(register)|symbolic name}
                  ,LENGTH={(register)|symbolic name}
                  ,RU={(register)|symbolic name}
                  ,RULENG={(register)|symbolic name}
                  ,DEST={(register)|symbolic name}
                  [,OPTION={NOCNM|NOPRID|SALT}]
                  [,TYPE={CHAIN|RU|NOEMBED}]
                  [,RTYPE={REPLY|RESPONSE}]
                  ,TARGET={(register)|symbolic name}
                  [,SECONDS={(register)|symbolic name}]
```

Note: SWB, DSRB, and at least one other parameter are required.

SWB
Register, or symbolic name of a fullword area, containing the address of a service work block (SWB) to be passed to the CNM interface service routine DSIZCSMM.

DSRB
Register, or symbolic name of a fullword area, containing the address of a data services request block (DSRB) to be passed to the CNM interface service routine DSIZCSMM.

INPUT
Register, or symbolic name of a fullword storage location, containing the address of a user input buffer. The size of the input buffer must be large enough to contain a 24-byte buffer header plus the length of the RU to be sent (if a Forward RU is to be built add an additional 28 bytes to the size of the input buffer). This buffer must contain a buffer header followed by text; it also holds the Deliver RU that is returned by the access method. To enable command processors or user exit routines that operate in 24-bit addressing mode to access the buffer, it must reside below 16 Mb. To receive a reply over the CNM interface, the buffer must accommodate at least a 32-byte reply (a 24-byte NetView buffer header and an eight-byte positive or negative response).

LENGTH
Register, or symbolic name of a fullword storage location, containing the length in binary of the input buffer.

RU
Register, or symbolic name of a fullword storage location, containing the address of a user area. The area is an RU that is to be embedded within the Forward RU.

RULENG
Register, or symbolic name of a fullword user area, containing the length in binary of the embedded RU buffer. The RULENG may not exceed 32743 decimal bytes.

DEST
Register, or symbolic name of a fullword user area, containing the address of the network destination to which the embedded RU is sent. This network destination must be eight characters long, left-justified, and padded with blanks if necessary.

Note: You may specify either OPTION or TYPE, but not both.

OPTION
Allows nonstandard Forward RUs to be sent by CNM interface services.

NOCNM
Indicates that a Forward RU will be sent that does not contain a CNM header, or procedure-related ID (PRID). An example is a forward RU containing NS IPL command types of INIT, TEXT, and FINAL.

Note: For RTYPE=REPLY, you may specify either OPTION=NOCNM or SECONDS, but not both.

NOPRID
Indicates that a Forward RU will be sent that does not contain a procedure-related ID (PRID) in the CNM header. An example is a REQMS that does not require a reply RECFMS because of user protocols.

Note: For RTYPE=REPLY, you may specify either OPTION=NOPRID or SECONDS, but not both.

SALT
Indicates that a Forward RU will be sent that is associated with a target. This flag alerts VTAM that a target to the SNA address list translation is required for an NMVT to be transported using a Forward RU.

TYPE
Controls the processing for the RU.

CHAIN
Indicates that the DSRB has received data and should remain in use to accept further RUs associated with the specific request. If TYPE=CHAIN is specified, the SWB and DSRB parameters are required; all other operands are invalid. This parameter is invalid with an unsolicited DSRB.

Note: You may specify either TYPE=CHAIN or SECONDS, but not both.

RU
Indicates that the input RU is not to be embedded in a Forward RU.

Note: You may specify either TYPE=RU or SECONDS, but not both.

NOEMBED
Indicates that a Forward RU is not to be built for this request.

RTYPE
Describes the response required for the request.

REPLY
Indicates that a Reply RU is required for completion of the request.

Note: You may specify only one of the following: OPTION=NOCNM, OPTION=NOPRID, or SECONDS.

RESPONSE
Indicates that a positive response is sufficient to complete the request.

TARGET
Register, or symbolic name of a fullword user area, containing the address of the network component that is the object of the embedded RU, or symbolic name of that network component. This network component must be eight characters long, left-justified, and padded with blanks if necessary.

SECONDS
Specifies the number of seconds to wait before cancelling the outstanding request. The number must have a positive value no greater than 86,400. If the SECONDS parameter is not specified or the value is zero, no timeout function is performed (an indefinite wait is possible). If you specify SECONDS, you cannot specify any of the following: TYPE=RU, TYPE=CHAIN, OPTION=NOCNM, nor OPTION=NOPRID. If SECONDS is specified and RTYPE=REPLY, then DEST must be specified.[5]

Usage Examples

1. DSIZCSMS SWB=SWBADDR, DSRB=DSRBADDR, INPUT=RAQDDR, LENGTH=FORWDLEN, RU=READYRU, RULENG=READLEN, DEST=DESTNAME, RTYPE=REPLY, SECONDS=TIMEOUT.

   The data will be sent across the CNM interface to DESTNAME. READYRU contains the request RU that is to be sent. The reply information is returned in the RQADDR buffer. After TIMEOUT seconds, the request will be cancelled. If an associated reply is received later, it will be discarded.

2. DSIZCSMS SWB=SWBADDR, DSRB=DSRBADDR, INPUT=RAQDDR, LENGTH=FORWDLEN, RU=READYRU, RULENG=READLEN, DEST=DESTNAME, RTYPE=REPLY.

   The data will be sent across the CNM interface to DESTNAME. READYRU contains the request RU that is to be sent. The reply information is returned in the RQADDR buffer. The request will remain outstanding until a reply is received.

Return Codes in Register 15

Major return codes in Register 15:

0	Function was successful; data was sent to VTAM.
4	The requested function could not be performed.
8	The input buffer was too small to build a Forward RU.
12	An error was found in parameter specification.
16	The program was not executing under a data services task.
20	The RULENG exceeded the maximum RU length allowed.

Minor return codes in Register 0:

0	The function was successful.

[5] The seconds function is available only with VTAM V3R1.1 or later.

4	The SWB was invalid.
8	The DSRB was invalid.
12	The DSRB that was passed was in use.
16	An unsolicited DSRB was passed.
20	An invalid operator ID was specified in the DSRB.
24	Reserved.
28	There was insufficient storage to process the request.
32	The CNM interface is inactive.
36	The request was rejected by the access method.
40	A user exit rejected the request.
44	Data truncation occurred during the user exit processing.
48	The specified SECONDS value was invalid.

Note: For major return codes of 8 or 20, register 0 will contain the RULENG.

DSIZVSMS – VSAM Data Services

Macro DSIZVSMS is a data services macro. It requests VSAM services for a data services command processor (DSCP).

DSIZVSMS provides access to VSAM services that perform I/O to the specified problem determination file or data set. The parameters allow access for data recording, data retrieval, and data deletion.

MVT addressability is not required.

```
[name] DSIZVSMS  SWB={(register)|symbolic name}
                 ,DSRB= {(register)|symbolic name}
                 ,FUNC={GET|PUT|POINT|ENDREQ|ERASE}
                 [,KEY={(register)|symbolic name}]
                 [,KEYLEN={(register)|symbolic name}]
                 [,OPTION=({SEQ|DIR|SKP},{ARD|LRD},{FWD|
                        BWD},{NUP|NSP|UPD},{KEQ|KGE},
                        {FKS|GEN})]
                 [,DATAREA={(register)|symbolic name}]
```

SWB
Register, or symbolic name of a fullword area, containing the address of a service work block (SWB). The SWB contains a save area, work area, and TIB address data. The caller must initialize the SWBTIB field in the SWB with a valid TIB address.

DSRB
Register, or symbolic name of a fullword, containing the address of a data services request block (DSRB). The DSRB contains request information such as RPL, ACB, ECB, and fields used by the DST VSAM service routine for VSAM I/O.

FUNC
Describes the VSAM request macro to be issued. See *OS/VS VSAM Programmer's Guide* (for MVS) and *Using VSE/VSAM Commands and Macros* (for VM and VSE only) for more information on how to specify FUNC.

KEY
Register containing the address of the VSAM key to be used for access to the requested data, or symbolic name of a fullword that contains the key.

KEYLEN
Register, or symbolic name of a fullword, containing the length in bytes of the key pointed to by KEY, or a symbolic name of a fullword with the length (in bytes) of the key. The default value is 1.

OPTION
Specifies the type of access to the file through requests defined by the NetView RPL. Options are arranged in groups. The first time the RPL is set up, you must specify one, and only one, option from each group. Subsequently, you may specify one option from one or more groups. You must separate multiple options with commas.

This parameter has no defaults. This parameter is not valid when FUNC=ERASE or FUNC=ENDREQ is specified. See *OS/VS VSAM Programmer's Guide* (for MVS) and *Using VSE/VSAM Commands and Macros* (for VM and VSE only) for more information on how to specify OPTION.

DATAREA
Register containing the address of a user work buffer, or symbolic name of that buffer. The buffer must be large enough to contain the maximum size record in the file or data set and is used by VSAM in the processing of records. This buffer must contain an initialized BUFHDR, followed by text. For MVS/XA, if any command processors or user exits which operate in 24-bit addressing mode are to access the data area, the data area must reside below 16 Mb.

Return Codes in Register 15

Major return codes in Register 15:

0 Successful completion of VSAM function.

4 Manipulative macro error occurred during processing. See the explanation of RPL feedback codes in *OS/VS VSAM Programmer's Guide* (for MVS) and *Using VSE/VSAM Messages and Codes* (for VM and VSE only).

8 An error occurred in the EXECUTE form of a manipulative macro. A parameter was not in the list. See the explanation of RPL feedback codes in *OS/VS VSAM Programmer's Guide* (for MVS) and *Using VSE/VSAM Messages and Codes* (for VM and VSE only).

12 Unsuccessful completion.

16 DSIZVSMS was issued while not executing under a DST.

Minor return codes in Register 0:

0 Successful completion.

4 The specified DSRB was invalid or in use.

8 An ACB was unavailable or was not open.

12 Resume verb processing error.

16 A user exit rejected the request.

20 The VSAM I/O request was invalid or there was an I/O scheduling error.

24 Data truncation occurred during substitution of data in a user exit. Or, control block storage could not be obtained.

28 A user exit returned an invalid return code.

Appendix

Appendix A. Assembler Samples

This appendix contains a table of the assembler samples that are shipped with NetView in SYS1.CNMSAMP. When data set names are referred to in this appendix, two names are given, such as ATMPPLT (CNMS4202). The first name is the alias name, and the name in parenthesis is the one in the NetView samples library. You can use either name to access the samples. DSICMD has definitions for the alias names to allow those names to be entered as commands.

The following steps allow you to enter the member names as commands:

1. Assemble and link-edit the samples using the alias name.
2. Delete the asterisk (*) in column 1 of the appropriate CMDMDL statment in DSICMD to be able to execute the alias name as a command. No entries are needed in DSICMD for user exits.
3. NetView must be recycled to pick up the DSICMD changes.

Notes:

1. See the prologues of the samples for information about how certain samples are related and special cases for user exit routines and other samples (DSIUSR00).
2. The alias name is the same as the CSECT name.
3. Each alias name for the assembler samples begins with the letter *A*, except for DSIUSR00 and DSIEX02A.

This appendix also contains a description of each sample and coded samples of a user exit routine and a command processor.

Assembler Samples Table

The following table refers to the assembler samples that are shipped with NetView. The table contains the function, the alias name, and the name of the member in SYS1.CNMSAMP.

Sample function	Assembler Alias	sample CNMSAMP
Template for assembler command processor	ATMPCMDP	CNMS4202
Defines an XITVN DST exit to initialize an empty VSAM data set	AXITVN	CNMS4270
Uses DSIMDS to build a message module	AMSGMOD	CNMS4271
Uses DSIWLS to write a message to the NetView log	AWRTLOG	CNMS4272
Uses DSIPSS for title line output	AMLWTO	CNMS4273
Uses DSIPSS to display date and time	ADATTIM	CNMS4274
Logs text to a sequential log	ASEQLOG	CNMS4275
Lists a member of DSIPARM	ALISTMEM	CNMS4276
User written optional subtask	AOPTTSK	CNMS4277
Uses DSIMBS to build messages	ABLDMSG	CNMS4278
Uses DSIPSS to display a full screen panel	APSSFULL	CNMS4279
Calls another command	ACALLCMD	CNMS4280
User defined message member	DSIUSR00	CNMS4281
Template for assembler user exit	ATMPUXIT	CNMS4282
Uses DSIEX02A to manipulate messages	DSIEX02A	CNMS4283

Assembler Samples Description

For each sample, a description of the function and the NetView service macros utilized are given.

ATMPCMDP (CNMS4202)

This sample is a template for command processors in assembler.

This sample is included in "Template for a Command Processor" on page 71.

AXITVN (CNMS4270)

This sample is an XITVN DST exit. It provides the initial record for an empty VSAM data set.

NetView service macros utilized in this sample: DSICBS, DSIDATIM.

AMSGMOD (CNMS4271)

This sample uses the message definition services (DSIMDS) to build a user-defined message module (AMSGMOD). It is used in conjunction with the ABLDMSG sample command processor and DSIEX02A.

NetView service macro utilized in this sample: DSIMDS.

AWRTLOG (CNMS4272)

This sample uses DSIWLS to write a message to the NetView log.

NetView service macros utilized in this sample: DSICBS, DSIDATIM, DSIWLS.

AMLWTO (CNMS4273)

This sample uses DSIPSS for title-line output.

NetView service macros utilized in this sample: DSICBS, DSIDATIM, DSIPSS.

ADATTIM (CNMS4274)

This sample uses DSIPSS to display date and time.

NetView service macros utilized in this sample: DSICBS, DSIDATIM, DSIPSS.

ASEQLOG (CNMS4275)

This sample logs text to a sequential log.

NetView service macros utilized in this sample: DSICBS, DSIDATIM, DSIMQS, DSIPSS, DSIWLS.

This sample is included in "Sample Command Processor for Sequential Logging" on page 238.

ALISTMEM (CNMS4276)

This sample reads and displays a member from the NetView DSIPARM data set. It also scope checks the supplied parm member name to prevent unauthorized display of DSIOPF.

NetView service macros utilized in this sample: DSICBS, DSIDATIM, DSIDKS, DSIKVS, DSIPSS.

AOPTTSK (CNMS4277)

This sample is an example of a user written optional subtask.

This sample is included in "Template for an Optional Task" on page 95.

ABLDMSG (CNMS4278)

This sample uses DSIMBS to build user defined messages.

NetView service macros utilized in this sample: DSICBS, DSIDATIM, DSIDEL, DSILOD, DSIMBS, DSIPSS.

APSSFULL (CNMS4279)

This sample uses DSIPSS to display a full screen panel, wait for terminal input, and echo the input.

NetView service macros utilized in this sample: DSICBS, DSIFRE, DSIGET, DSIPSS, DSIWAT.

ACALLCMD (CNMS4280)

This sample calls another command.

NetView service macros utilized in this sample: DSICBS, DSICES, DSIDATIM, DSIFRE, DSIGET, DSILCS, DSIPRS.

DSIUSR00 (CNMS4281)

This sample is an example of a user defined message member. It is used in conjunction with the ABLDMSG sample command processor.

ATMPUXIT (CNMS4282)

This sample is a template for assembler user exits. See "Template for a User Exit Routine" on page 48.

DSIEX02A (CNMS4283)

This sample is used to manipulate messages. The user exit is invoked for standard output to the operator's terminal. If DWO403I is the incoming message, it will MQS a start task request to start the task specified in DWO403I and swap the message buffer to indicate that the task has been started and that the request should be reissued.

NetView service macros utilized in this sample: DSICBS, DSIDEL, DSIFRE, DSIGET, DSILCS, DSILOD, DSIMBS, DSIMQS, DSIPRS.

This sample is included in "Sample User Exit" on page 229.

Assembler Coded Samples

This section contains an example of a user exit routine and a command processor.

Sample User Exit

This sample is an example of user exit DSIEX02A.

```
DSIEX02A CSECT
***********************************************************************
*                                                                     *
* IEBCOPY   SELECT MEMBER=((CNMS4283,DSIEX02A,R))                     *
*                                                                     *
* (C) COPYRIGHT IBM CORP. 1989                                        *
*                                                                     *
* MODULE NAME: DSIEX02A                                               *
*                                                                     *
* FUNCTION: THIS USER EXIT IS INVOKED FOR STANDARD OUTPUT TO THE      *
*           OPERATOR'S TERMINAL (DSIPSS TYPE=OUTPUT, IMMED OR FLASH). *
*           IF MESSAGE DWO403I IS THE INCOMING MESSAGE, IT WILL MQS   *
*           A START TASK REQUEST TO START THE SPECIFIED OPTIONAL      *
*           TASK AND SWAP THE MESSAGE BUFFER TO INDICATE THAT THE     *
*           TASK HAS BEEN STARTED AND TO RE-ISSUE THE REQUEST.        *
*                                                                     *
*                                                                     *
* INSTALLATION:  AMSGMOD MESSAGE TABLE (CNMS4271) MUST BE LINKED      *
*                INTO THE USER LIBRARY.                               *
*                                                                     *
*                                                                     *
***********************************************************************
* INPUT:   REG 1 - ADDRESS OF USER EXIT PARAMETER LIST (DSIUSE)       *
*          REG13 - ADDRESS OF CALLER'S SAVE AREA                      *
*          REG14 - RETURN ADDRESS                                     *
*          REG15 - ENTRY ADDRESS                                      *
*                                                                     *
* OUTPUT:                                                             *
*                                                                     *
* OUTPUT:   REG0  - ADDRESS OF SWAPPED MESSAGE IF REG15 = 8           *
*                                                                     *
*           REG15 - RETURN CODE                                       *
*               0 = USE MESSAGE AS IS, DO NOT DELETE OR REPLACE       *
*               4 = DELETE MESSAGE FROM THE TERMINAL & THE LOG        *
*               8 = REPLACE MESSAGE WITH MESSAGE ADDRESSED IN REG0    *
*                                                                     *
* NETVIEW MACROS:                                                     *
*                                                                     *
*          DSICBS   - CONTROL BLOCK SERVICE                           *
*          DSIDEL   - DELETE USER-DEFINED MODULE                      *
*          DSIFRE   - FREEMAIN STORAGE SERVICE                        *
*          DSIGET   - GETMAIN STORAGE SERVICE                         *
*          DSILCS   - LOCATE CONTROL BLOCK                            *
*          DSILOD   - LOAD USER-DEFINED MODULE                        *
*          DSIMBS   - MESSAGE BUILD SERVICE                           *
*          DSIMQS   - MESSAGE QUEUEING SERVICE                        *
*          DSIPRS   - PARSE DESCRIPTOR BLOCK                          *
***********************************************************************
         EJECT
*
***********************************************************************
*
*        INCLUDE THE REQUIRED CONTROL BLOCKS
*
***********************************************************************
*
         DSICBS DSITIB,DSITVB,DSIMVT,DSISVL,DSISWB,DSIUSE,DSIIFR,      X
```

```
               DSISVL,DSIPDB,PRINT=NO
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7                     MVT
R8       EQU   8                     TVB
R9       EQU   9                     USE
R10      EQU   10
R11      EQU   11
R12      EQU   12                    BASE REG
R13      EQU   13                    SAVEAREA
R14      EQU   14
R15      EQU   15
         EJECT
*
***********************************************************************
*                                                                     *
*  DSIEX02:  SET UP ENTRY LINKAGE                                     *
*                                                                     *
***********************************************************************
*
DSIEX02A CSECT
         USING *,R15
         B     PROLOG
         DC    C'DSIEX02A &SYSDATE. AT &SYSTIME.'
PROLOG   DS    0H
         STM   R14,R12,12(R13)       SAVE CALLER'S REGISTERS
         DROP  R15
         LR    R12,R15               SAVE BASE REGISTER
         USING DSIEX02A,R12          REG 12 IS THE BASE REG
         USING DSIUSE,R1             REG 1 POINTS TO DSIUSE
         L     R11,USERSWB           LOAD REG 11 WITH SWB ADDRESS
         USING DSISWB,R11            BASE SWB
         LA    R2,SWBSAVEA           GET ADDRESS OF OUR SAVE AREA
         ST    R2,8(,R13)            SET CALLER'S FORWARD POINTER
         ST    R13,4(,R2)            SET OUR BACKWARD POINTER
         LR    R13,R2                REG 13 CONTAINS SAVE AREA ADDR
         LR    R9,R1                 MOVE DSIUSE ADDRESS
         DROP  R1                    DROP ORIGINAL DSIUSE BASING
         USING DSIUSE,R9             REG 9 POINTS TO DSIUSE
         L     R8,USERTVB            LOAD REG 8 WITH TVB ADDRESS
         USING DSITVB,R8             BASE THE TVB
         L     R7,TVBMVT             LOAD REG 8 WITH TVB ADDRESS
         USING DSIMVT,R7             BASE THE MVT
*
***********************************************************************
*                                                                     *
*  MESSAGE DWO403I IS NOT EXPECTED IN AN EXIT.  CHECK TVBINXIT AND    *
*  EXIT IF SET.                                                       *
*                                                                     *
***********************************************************************
*
         TM    TVBIND3,TVBINXIT      IS TVBINXIT ON
         BO    ASIS                  YES, LEAVE
*
***********************************************************************
*                                                                     *
*  CHECK INPUT MESSAGE NUMBER AND EXIT IF IT IS NOT "DWO403I"         *
*                                                                     *
***********************************************************************
*
         L     R4,USERMSG            LOAD REG 4 WITH INPUT AIFR
         USING  BUFHDR,R4              BASE THE INPUT MESSAGE BUFFER
         L      R2,USERMSG             GET ADDRESS OF AIFR
         AH     R2,HDRTDISP            LOCATE START OF TEXT
         DROP   R4
         USING  DSIIFR,R2              BASE THE IFR
         L      R3,IFRAUTBA            LOAD REG 3 WITH ADDR INPUT MSG
         USING  BUFHDR,R3
         L      R4,IFRAUTBA
         AH     R4,HDRTDISP            LOCATE START OF TEXT
         CLC    0(7,R4),=CL7'DW0403I'  IS IT DW0403I
         BNE    ASIS                   NO, EXIT
*
***********************************************************************
*                                                                     *
*   IF MSG DW0403I IS NOT A NETVIEW SINGLE LINE MESSAGE OR            *
*                  IS A COPY OF ANOTHER MESSAGE OR                    *
*                  IS A MESSAGE FROM ANOTHER DOMAIN, THEN EXIT.       *
*                                                                     *
***********************************************************************
*
         CLI    HDRMTYPE,HDRTYPEN      NETVIEW SINGLE LINE MSG ??
         BNE    ASIS                   NO, COULD BE "COMMAND" ECHO
         TM     IFRAUIN2,IFRAUSEC      WAS IT ROUTED USING ASSIGN=SEC
         BO     ASIS                   YES, LEAVE
         TM     IFRAUIN2,IFRAUCPY      WAS IT ROUTED USING ASSIGN=CPY
         BO     ASIS                   YES, LEAVE
         TM     IFRAUIND,IFRAUXDM      IS IT FROM ANOTHER DOMAIN
         BO     ASIS                   YES, LEAVE
         ST     R3,INPUTMSG            SAVE ADDR OF INPUT MESSAGE
         DROP   R3
*
***********************************************************************
*                                                                     *
*   INITIALIZE ALL POINTERS TO ZERO                                   *
*                                                                     *
***********************************************************************
*
         SR     R4,R4
         ST     R4,MSGBFR              INIT MSG BUFFER POINTER
         ST     R4,ADDRSWB             INIT SWB POINTER
         ST     R4,ADDRPDB             INIT PDB POINTER
         ST     R4,MSGTABLE            INIT MSG TABLE POINTER
         MVC    TASKID,=8C' '          INIT TASK_ID FIELD
*
***********************************************************************
*                                                                     *
*   GET A BUFFER FOR ISSUING/SWAPPING MESSAGES                        *
*                                                                     *
***********************************************************************
*
         LA     R4,BFRLENG             GET LENGTH OF MSG BUFFER
         DSIGET LV=(R4),                                               X
               A=MSGBFR,                                               X
               Q=NO,                                                   X
               TASKA=(R8),                                             X
               SP=0
         LTR    R15,R15                CHECK IF DSIGET SUCCESSFUL
         BNZ    ASIS                   DSIGET UNSUCCESSFUL
         L      R3,MSGBFR
         USING  BUFHDR,R3
         STH    R4,HDRBLENG            SET HDRBLENG IN BUFFER
         DROP   R3
*
***********************************************************************
*                                                                     *
*   LOAD THE USER MESSAGE TABLE                                       *
*                                                                     *
***********************************************************************
*
         DSILOD EP=AMSGMOD            LOAD THE MESSAGE TABLE
         LTR    R15,R15               WAS DSILOD SUCCESSFUL
         BNZ    LODFAIL               NO, EXIT
         ST     R0,MSGTABLE
*
***********************************************************************
*                                                                     *
*   GET ANOTHER SWB IN ORDER TO ISSUE NETVIEW SERVICE MACROS          *
*                                                                     *
***********************************************************************
*
         DSILCS CBADDR=ADDRSWB,                                        X
               SWB=GET                GET A NEW SWB
         LTR    R15,R15               TEST DSILCS RETURN CODE
         BNZ    NOSWB                 NOTIFY USER AND EXIT
         L      R5,ADDRSWB            PUT THE NEW SWB ADDR IN REG 5
         L      R4,TVBTIB             PUT THE TIB ADDR IN REG 4
         ST     R4,SWBTIB-DSISWB(,R5) STORE TIB ADDR IN THE NEW SWB
*
***********************************************************************
*                                                                     *
* ISSUE DSIPRS TO DETERMINE THE SIZE OF THE PARSE TABLE REQUIRED TO   *
* PARSE THE INPUT BUFFER.                                             *
*                                                                     *
***********************************************************************
*
         DSIPRS SWB=ADDRSWB,                                           X
               BFR=INPUTMSG,                                           X
               PDBSIZE=SIZEPDB
         LTR    R15,R15               WAS DSIPRS SUCCESSFUL
         BZ     GETPDB                YES, CONTINUE
         MVC    REQUEST,PRS           NO, SET MACRO NAME
         B      MACFAIL               NOTIFY USER AND EXIT
         SPACE
*
***********************************************************************
*                                                                     *
* ISSUE DSIGET TO GET STORAGE FOR THE PARSE TABLE                     *
*                                                                     *
***********************************************************************
*
GETPDB   EQU    *
         L      R4,SIZEPDB            GET TABLE SIZE
         DSIGET LV=(R4),                                               X
               A=ADDRPDB,                                              X
               Q=YES,                                                  X
               TASKA=(R8),                                             X
               SP=0
         LTR    R15,R15               CHECK IF DSIGET SUCCESSFUL
         BZ     INITBFR               DSIGET SUCCESSFUL
         SR     R2,R2
         ST     R2,INPUTMSG           ZERO POINTER TO INPUT MSG
         BAL    R6,FREESTOR           FREE THE STORAGE AND EXIT
         B      ASIS
*
***********************************************************************
*                                                                     *
* INITIALIZE THE CONTROL BLOCK HEADER IN THE PARSE TABLE              *
*                                                                     *
***********************************************************************
*
INITBFR  EQU    *
         L      R5,ADDRPDB            ADDRESS OF THE PARSE TABLE
         USING  DSICBH,R5              ADDRESSABILITY TO PARSE TABLE
         STH    R4,CBHLENG             SET THE CONTROL BLOCK LENGTH
         MVI    CBHID,CBHPDB           SET THE CONTROL BLOCK ID = PDB
         DROP   R5
*
         EJECT
*
***********************************************************************
*                                                                     *
*  PARSE THE INPUT MESSAGE                                            *
*                                                                     *
***********************************************************************
*
         L      R2,ADDRPDB             GET PARSE TABLE ADDRESS
         DSIPRS SWB=ADDRSWB,                                           X
               BFR=INPUTMSG,                                           X
               PDB=(R2)
         LTR    R15,R15                CHECK IF DSIPRS SUCCESSFUL
         BZ     GETNTRY                YES, CONTINUE
         MVC    REQUEST,PRS            NO, SET MACRO NAME
         B      MACFAIL                NOTIFY USER AND EXIT
         EJECT
*
***********************************************************************
*                                                                     *
*  FIND PDBENTRY THAT CONTAINS THE TASK NAME.                         *
*                                                                     *
***********************************************************************
*
GETNTRY  EQU    *
         L      R5,ADDRPDB
         USING  DSIPDB,R5
         LA     R3,2                   SET FOR ENTRY NUMBER OF TASKID
         LA     R2,PDBTABLE            GET ADDR OF 1ST ENTRY
         USING  PDBENTRY,R2
         DROP   R5
LOOP1    EQU    *
         LA     R2,PDBENTND-PDBENTRY(,R2)  GET NEXT ENTRY
         BCT    R3,LOOP1
*
***********************************************************************
*                                                                     *
*  BUILD IFRCODE_3 AND MQS TO START THE TASK                          *
*                                                                     *
***********************************************************************
*
         L      R4,MSGBFR              LOAD REG 4 WITH MSG BFR ADDR
         USING  BUFHDR,R4              BASE THE MESSAGE BUFFER
         LA     R6,HDRMSGLN            LOAD REG 4 WITH OFFSET TO TEXT
         STH    R6,HDRTDISP            SET HDRTDISP
         MVC    HDRDOMID(8),MVTCURAN   SET DOMAIN ID
         MVC    HDRSENDR(8),TVBOPID    SET SENDER ID
         MVI    HDRMTYPE,HDRTYPEI      MSG TYPE IS IFR
         L      R5,MSGBFR              GET ADDRESS OF MSG BUFFER
         AH     R5,HDRTDISP            GET ADDRESSABILITY TO IFR
         USING  DSIIFR,R5
         LA     R6,IFRCODCR
         STH    R6,IFRCODE             SET BUFFER AS A COMMAND
         L      R3,INPUTMSG            GET INPUT MESSAGE
         AH     R3,PDBDISP             LOCATE THE TASK NAME
         SR     R10,R10
         ICM    R10,B'0001',PDBLENG    GET LENGTH OF TASK NAME
         ST     R10,LTASKID            SET LENGTH OF TASK NAME
         BCTR   R10,0                  DECREMENT LENGTH FOR MOVE
         EX     R10,MOVE1              GET TASK NAME FROM INPUT BFR
         MVC    IFRPARMS(L'CMD),CMD    SET COMMAND
         EX     R10,MOVE2              SET TASK NAME IN MSG BFR
         LA     R10,1(,R10)            INCREMENT FOR ACTUAL LENGTH
         LA     R6,IFRPARMS-DSIIFR     LOAD REG 4 WITH LENGTH OF IFR
         LA     R6,L'CMD(,R6)          ADD LENGTH OF COMMAND
         LA     R6,0(R10,R6)           ADD LENGTH OF TASK NAME
         STH    R6,HDRMLENG            SET HDRMLENG
         DSIMQS TASKID=TVBOPID,                                        X
               BFR=(R4),                                               X
               SWB=ADDRSWB             SEND TO OPER
         LTR    R15,R15                WAS DSIMQS SUCCESSFUL
         BZ     STARTED                YES, CONTINUE
         MVC    REQUEST,MQS            NO, SET MACRO NAME
         B      MACFAIL                NOTIFY USER AND EXIT
         SPACE
MOVE1    MVC    TASKID(0),0(R3)        GET THE TASK NAME
MOVE2    MVC    IFRPARMS+L'CMD(0),TASKID  SET TASK NAME
         DROP   R2,R5
         EJECT
*
***********************************************************************
*                                                                     *
*  STARTED:  SWAP THE TEXT IN MESSAGE BUFFER TO INDICATE THAT WE      *
*            HAVE STARTED THE TASK AND TO RETRY THE REQUEST.          *
*                                                                     *
***********************************************************************
*
STARTED  EQU    *
         L      R3,INPUTMSG            GET INPUT MESSAGE
         MVC    HDRMLENG(HDRMSGLN),0(R3)   SET THE MESSAGE LENGTH
         MVI    HDRMTYPE,HDRTYPEU      SET MSG TYPE TO USER
         LA     R2,BFRLENG
         STH    R2,HDRBLENG            SET THE BUFFER LENGTH
         DSIMBS SWB=ADDRSWB,                                           X
               MID=004,                                                X
               BFR=(R4),                                               X
               MSGTBL=MSGTABLE,                                        X
               P1=(TASKID,LTASKID)     BUILD MSG004I
         DROP   R4
         B      SWAP
*
***********************************************************************
*                                                                     *
*  NOSWB:  TO SWAP INPUT MESSAGE FOR "NO SWB" MESSAGE                 *
*                                                                     *
***********************************************************************
*
NOSWB    EQU    *
         L      R4,MSGBFR              LOAD REG 4 WITH MESSAGE BFR ADDR
         USING  BUFHDR,R4              BASE THE MESSAGE BUFFER
         L      R3,INPUTMSG            GET THE INPUT MSG ADDRESS
         MVC    HDRMLENG(HDRMSGLN),0(R3)  COPY INPUT BUFHDR TO MSG BFR
         MVI    HDRMTYPE,HDRTYPEU      MSG TYPE IS USER
         LA     R2,BFRLENG
         STH    R2,HDRBLENG            SET THE BUFFER LENGTH
         LA     R2,L'SWBMSG
         STH    R2,HDRMLENG            SET THE MESSAGE LENGTH
         L      R2,MSGBFR              GET ADDRESS OF MESSAGE BFR
         AH     R2,HDRTDISP            LOCATE START OF TEXT
         MVC    0(L'SWBMSG,R2),SWBMSG  PUT MESSASGE IN BUFFER
         DROP   R4
         B      SWAP
         SPACE
*
***********************************************************************
*                                                                     *
*  FREESTOR:  TO FREE ALL STORAGE GOTTEN                              *
*                                                                     *
***********************************************************************
*
FREESTOR EQU   *
         L     R2,ADDRSWB               GET SWB POINTER
         LTR   R2,R2                    IS THERE AN SWB
         BZ    CK4NPUT                  NO, CHECK FOR INPUT MSG BFR
         DSILCS CBADDR=ADDRSWB,                                        X
               SWB=FREE                 FREE THE SWB
CK4NPUT  EQU   *
         L     R3,INPUTMSG              GET THE INPUT MSG POINTER
         LTR   R3,R3                    DOES IT NEED FREEING
         BZ    CK4PDB                   NO, CHECK FOR A PDB
         USING BUFHDR,R3
         LH    R2,HDRBLENG              GET LENGTH OF INPUT MSG BUFFER
         DROP  R3
         DSIFRE A=INPUTMSG,                                            X
               LV=(R2),                                                X
               Q=NO,                                                   X
               SP=0                     FREE THE INPUT MSG BUFFER
CK4PDB   EQU   *
         L     R2,ADDRPDB               GET THE PDB POINTER
         LTR   R2,R2                    IS THERE ONE TO FREE
         BZ    CK4MSGM                  NO, CHECK FOR A MESSAGE TABLE
         L     R2,SIZEPDB               GET THE SIZE OF THE PDB
         DSIFRE A=ADDRPDB,                                             X
               TASKA=(R8),                                             X
               Q=YES,                                                  X
               SP=0                     FREE THE PDB
CK4MSGM  EQU   *
         L     R2,MSGTABLE              GET THE MESSAGE TABLE POINTER
         LTR   R2,R2                    IS THERE ONE TO FREE
         BZ    XITFREE                  NO, RETURN TO CALLER
         DSIDEL EP=AMSGMOD              YES, DELETE THE MESSAGE TABLE
XITFREE  EQU   *
         BR    R6                       RETURN TO CALLER
*
***********************************************************************
*                                                                     *
*  MACFAIL:  TO SWAP INPUT MSG TO INDICATE DSIPRS/DSIMQS FAILED       *
*                                                                     *
***********************************************************************
*
MACFAIL  EQU   *
         L     R4,MSGBFR                LOAD REG 4 WITH MESSAGE BFR ADDR
         USING BUFHDR,R4                BASE THE MESSAGE BUFFER
         L     R3,INPUTMSG              GET THE INPUT MESSAGE BUFFER
         MVC   HDRMLENG(HDRMSGLN),0(R3)  COPY INPUT BUFHDR TO MSG BFR
         MVI   HDRMTYPE,HDRTYPEU        SET MSG TYPE TO USER
         LA    R2,BFRLENG               GET THE MSG BUFFER LENGTH
         STH   R2,HDRBLENG              SET MSG BFR LENGTH IN MSG BFR
         DSIMBS SWB=ADDRSWB,                                           X
               MID=005,                                                X
               BFR=(R4),                                               X
               P1=(REQUEST,6),                                         X
               MSGTBL=MSGTABLE          BUILD MSG005I
         DROP  R4
         B     SWAP
         SPACE
*
***********************************************************************
*                                                                     *
*  LODFAIL:  TO SWAP INPUT MSG TO INDICATE DSILOD FAILED              *
*                                                                     *
***********************************************************************
```

```
*
LODFAIL  EQU    *
         L      R4,MSGBFR              LOAD REG 4 WITH MESSAGE BFR ADDR
         USING  BUFHDR,R4              BASE THE MESSAGE BUFFER
         L      R3,INPUTMSG            GET INPUT MESSAGE BUFFER ADDRESS
         MVC    HDRMLENG(HDRMSGLN),0(R3)  SET MSG LENGTH IN MSG BUFFER
         MVI    HDRMTYPE,HDRTYPEU      SET MSG TYPE TO USER
         LA     R2,BFRLENG             GET THE MSG BUFFER LENGTH
         STH    R2,HDRBLENG            SET THE MSG BUFFER LENGTH
         LA     R2,L'LODMSG            GET MESSAGE LENGTH
         STH    R2,HDRMLENG            SET MSG LENGTH IN MSG BUFFER
         L      R2,MSGBFR              GET ADDRESS OF MESSAGE BFR
         AH     R2,HDRTDISP            LOCATE START OF TEXT
         MVC    0(L'LODMSG,R2),LODMSG  PUT MESSASGE IN BUFFER
         DROP   R4
         B      SWAP
         SPACE
         EJECT
*
***********************************************************************
*                                                                     *
*  ASIS:  TO PROCESS THE BUFFER AS IT IS AND RETURN                   *
*                                                                     *
***********************************************************************
*
ASIS     EQU   *
         LA    R15,USERASIS            SET AN ASIS RETURN CODE
         L     R13,4(,R13)             RESTORE CALLER'S SAVE AREA ADDR
         L     R14,12(,R13)            RESTORE CALLER'S REG 14
         LM    R0,R12,20(R13)          RESTORE CALLER'S REGS 0-12
         BR    R14                     RETURN TO CALLER
         SPACE
*
***********************************************************************
*                                                                     *
*  SWAP:  TO SWAP A BUFFER FOR THE INPUT BUFFER AND RETURN            *
*                                                                     *
***********************************************************************
*
SWAP     EQU   *
         L      R4,USERMSG             GET ADDRESS OF AIFR
         USING  BUFHDR,R4
         L      R2,USERMSG             GET ADDRESS OF AIFR
         AH     R2,HDRTDISP            LOCATE START OF TEXT
         USING  DSIIFR,R2
         L      R3,MSGBFR              GET ADDRESS OF MSG BUFFER
         ST     R3,IFRAUTBA            PUT NEW MSG IN AIFR
         ST     R3,IFRAUTBL            SET LAST MSG OF AIFR CHAIN
         DROP   R2,R4
         BAL    R6,FREESTOR            FREE THE STORAGE
         LA     R15,USERSWAP           SET A SWAP RETURN CODE
         L      R0,USERMSG             LOAD REG 0 WITH ADDR OF IFRCODAI
         L      R13,4(,R13)            RESTORE CALLER'S SAVE AREA ADDR
         L      R14,12(,R13)           RESTORE CALLER'S REG 14
         LM     R1,R12,24(R13)         RESTORE CALLER'S REGS 1-12
         BR     R14                    RETURN TO CALLER
         SPACE
         EJECT
         LTORG
*
***********************************************************************
*  CONSTANTS                                                          *
***********************************************************************
*
CMD      DC     C'START TASK='         START COMMAND
PRS      DC     C'DSIPRS'              PARSE MACRO
```

```
MQS      DC      C'DSIMQS'             MQS MACRO
LODMSG   DC      C'REQUEST CANNOT BE HONORED. LOAD FAILURE FOR AMSGMOD'
SWBMSG   DC      C'REQUEST CANNOT BE HONORED. UNABLE TO OBTAIN AN SWB'
*
***********************************************************************
*  DECLARES AND DSECTS                                                *
***********************************************************************
*
DSISWB   DSECT ,                      EXTEND THE SWB DEFINITION
         ORG   SWBADATD               POINT TO THE 256 BYTE WORK AREA
ADDRSWB  DS    A                      ADDRESS OF SWB
MSGBFR   DS    A                      ADDRESS OF MESSAGE BUFFER
ADDRPDB  DS    A                      ADDRESS OF PDB
MSGTABLE DS    A                      ADDRESS OF MESSAGE TABLE
INPUTMSG DS    A                      ADDRESS OF INPUT MESSAGE BUFFER
SIZEPDB  DS    F                      LENGTH OF PDB TO OBTAIN
LTASKID  DS    F                      LENGTH OF TASK NAME
REQUEST  DS    CL6                    LENGTH OF PDB TO OBTAIN
TASKID   DS    CL8                    TASK NAME
BFRLENG  EQU   100                    LENGTH OF MESSAGE BUFFER
         DS    XL(256-(*-SWBADATD))   ENSURE AUTODATA <= 256 BYTES
         SPACE
DSIEX02A CSECT ,                      RESUME CSECT
         END   DSIEX02A               END OF USER EXIT
```

Sample Command Processor for Sequential Logging

This sample is an example of a command processor to log text to a sequential log.

```
         TITLE 'ASEQLOG - WRITE TO A SEQUENTIAL LOG'
***********************************************************************
*                                                                     *
*  IEBCOPY   SELECT MEMBER=((CNMS4275,ASEQLOG,R))                     *
*                                                                     *
*  (C) COPYRIGHT IBM CORP. 1989                                       *
*                                                                     *
*                                                                     *
*  MODULE NAME: ASEQLOG                                               *
*                                                                     *
*  FUNCTION = THIS IS A COMMAND PROCESSOR FOR LOGGING                 *
*             TEXT TO A SEQUENTIAL LOG                                *
*                                                                     *
***********************************************************************
*                                                                     *
*  SYNTAX     ASEQLOG  TEXT-TO-LOG                                    *
*                                                                     *
***********************************************************************
*                                                                     *
*  INSTALLATION:                                                      *
*                                                                     *
*     (1)  ASSEMBLE AND LINKEDIT THIS MODULE AMODE=31,RMODE=ANY       *
*          TYPE=RENT                                                  *
*     (2)  ALLOC PRIMARY AND SECONDARY SEQUENTIAL DATA SET            *
*     (3)  USE DD NAMES IN NETVIEW PROC OR THE ALLOCATE COMMAND       *
*          TO ALLOCATE THE DATA SETS TO NETVIEW.                      *
*          ALLOCATE THE DATA SETS AS                                  *
*              SQLOGP & SQLOGS                                        *
*     (4)  ADD THE FOLLOWING STATEMENT TO DSIDMN                      *
*         TASK   MOD=DSIZDST,TSKID=SQLOGTSK,MEM=SQLOGMEM,PRI=3,INIT=Y *
*     (5)  ADD THE FOLLOWING MEMBER (SQLOGMEM) TO DSIPARM             *
*           DSTINIT FUNCT=OTHER,DSRBO=1                               *
*           DSTINIT PBSDN=SQLOGP                                      *
*           DSTINIT SBSDN=SQLOGS                                      *
*           LOGINIT AUTOFLIP=YES,RESUME=NO                            *
*     (6)  ADD THE FOLLOWING CMDMDL TO DSICMD                         *
*           ASEQLOG CMDMDL MOD=ASEQLOG,TYPE=R,RES=N                   *
*                                                                     *
*                                                                     *
***********************************************************************
*                                                                     *
*  INPUT:                                                             *
*      REGISTERS:                                                     *
*           R1  = DSICWB ADDRESS                                      *
*           R13 = ADDRESS OF STANDARD SAVEAREA                        *
*           R14 = RETURN ADDRESS                                      *
*           R15 = ENTRY POINT OF ROUTINE                              *
*                                                                     *
*                                                                     *
*  OUTPUT:                                                            *
*      REGISTERS:                                                     *
*           R0 - R14 = RESTORED UPON RETURN                           *
*                                                                     *
*           R15      = RETURN CODE - 0 IF ALL GOES WELL.  4 IF SOME   *
*                        KIND OF FAILURE (STORAGE REQUEST OR MESSAGE  *
*                        QUEUING) OCCURS.                             *
*                                                                     *
*                                                                     *
*                                                                     *
*  NETVIEW MACROS:                                                    *
*                                                                     *
*           DSICBS    - CONTROL BLOCK SERVICE                         *
*           DSIMQS    - MESSAGE QUEUEING SERVICE                      *
*           DSIPSS    - PRESENTATION SERVICES                        *
*           DSIWLS    - WRITE TO LOG SERVICE                         *
*                                                                    *
*                                                                    *
**********************************************************************
         EJECT
ASEQLOG  CSECT
         DSICBS DSITIB,DSITVB,DSIMVT,DSISVL,DSISWB,DSICWB,DSIPDB,PRINT=X
               NO
**********************************************************************
*  MODULE ADDRESSABILITY IS ESTABLISHED                              *
*  CALLER'S REGS ARE SAVED                                           *
**********************************************************************
         USING *,R15
         B     ENTLINK
         DC    CL8'ASEQLOG'
         DC    C'&SYSDATE'
ENTLINK  DS    0H
         STM   R14,R12,12(R13)     SAVE CALLERS REGS
         LR    R12,R15
         DROP  R15
         USING ASEQLOG,R12
**********************************************************************
* SET UP ADDRESSABILITY TO REQUIRED CONTROL BLOCKS.                  *
**********************************************************************
         USING DSICWB,R2           SET UP CWB ADDRESSABILITY
         USING DSITVB,R3           SET UP TVB ADDRESSABILITY
         USING DSITIB,R4           SET UP TIB ADDRESSABILITY
         USING DSIMVT,R5           SET UP MVT ADDRESSABILITY
         LR    R2,R1               PUT ADDRESS OF CWB IN ITS BASE REG.
         L     R4,CWBTIB           PUT ADDRESS OF TIB IN ITS BASE REG.
         L     R3,TIBTVB           PUT ADDRESS OF TVB IN ITS BASE REG.
         L     R5,TVBMVT           PUT ADDRESS OF MVT IN ITS BASE REG.
**********************************************************************
* CHAIN SAVE AREAS.                                                  *
**********************************************************************
         LR    R11,R13             MOVE ADDRESS OF CALLER SAVEAREA
         LA    R13,CWBSAVEA        PUT SAVEAREA AREA ADDRESS IN REG 13
         XC    8(4,R13),8(R13)     CLEAR MY FORWARD POINTER
         ST    R13,8(R11)          SET CALLERS FORWARD POINTER
         ST    R11,4(R13)          SET MY BACKWARD POINTER
         SLR   R15,R15             INITIALLY, RETURN CODE IS ZERO
         XC    CWBADATD(L'CWBADATD),CWBADATD CLEAR AUTODATA AREA
**********************************************************************
* INITIALIZE BUFFER THAT IS TO BE USED FOR MESSAGES.                 *
* MESSAGE LENGTH (HDRMLENG) will be set when the message to be sent  *
* HAS BEEN DETERMINED.                                               *
**********************************************************************
         LA    R6,MSGBFR              ADDRESS MESSAGE BUFFER
         LA    R7,L'MSGBFR            LENGTH OF MESSAGE BUFFER
         STH   R7,HDRBLENG-BUFHDR(R6) PUT BUFFER LENGTH IN HEADER
         LA    R7,BUFHDRND-BUFHDR     LENGTH OF HEADER (W/O EXTENSION)
         STH   R7,HDRTDISP-BUFHDR(R6) STORE DISPLACEMENT TO TEXT AREA
         MVC   HDRDOMID-BUFHDR(L'HDRDOMID,R6),MVTCURAN DOMAIN ID
         MVI   HDRMTYPE-BUFHDR(R6),HDRTYPEU USER MESSAGE TYPE
**********************************************************************
* CHECK RUNNING ENVIRONMENT.  ENSURE THAT THE COMMAND ENVIRONMENT     *
* IS OST, PPT, OR NNT. THIS WAY, THE PDB THAT WE EXPECT WILL EXIST.   *
**********************************************************************
         CLI   CBHTYPE-DSICBH(R3),CBHOST CHECK FOR OST
         BE    ENVIROK                ENVIRONMENT OKAY IF OST
         CLI   CBHTYPE-DSICBH(R3),CBHNNT CHECK FOR NNT
         BE    ENVIROK                ENVIRONMENT OKAY IF NNT
         CLI   CBHTYPE-DSICBH(R3),CBHPPT CHECK FOR NNT
         BE    ENVIROK                ENVIRONMENT OKAY IF PPT
*                                  SET UP INVALID TASK MESSAGE
```

```
         LR    R8,R6              COPY BUFFER ADDRESS
         AH    R8,HDRTDISP-BUFHDR(R6) POINT TO TEXT AREA
         LA    R9,INVTASK         ADDRESS INVALID TASK MESSAGE
         LA    R10,L'INVTASK      LENGTH OF INVALID TASK MESSAGE
         LR    R11,R10            SAVE THIS LENGTH
         BCTR  R10,0              DECREMENT FOR EXECUTE
         EX    R10,COPYMSG        COPY MESSAGE TEXT INTO BUFFER
         ALR   R8,R11             POINT TO POSITION AFTER MESSAGE
         LA    R9,TVBOPID         ADDRESS TASKID (OPID)
         LA    R10,L'TVBOPID      LENGTH OF TASKID (OPID)
         LR    R14,R10            COPY TVBOPID LENGTH
         BCTR  R10,0              DECREMENT FOR EXECUTE
         EX    R10,COPYMSG        COPY MESSAGE TEXT INTO BUFFER
         ALR   R11,R14            CALCULATE TOTAL MESSAGE LENGTH
         STH   R11,HDRMLENG-BUFHDR(R6) STORE MESSAGE LENGTH IN HEADER
         BAL   R14,SENDMSG        SEND MESSAGE
*                                 IF THE MESSAGE SEND FAILS, JUST
*                                 LEAVE WITH BAD RETURN CODE - WE
*                                 TRIED.
         LA    R15,ERRCODE        ERROR RETURN CODE
         B     EXIT               LEAVE THE MODULE
ENVIROK  EQU   *
***********************************************************************
* BUILD THE LOG HEADER                                                *
*       LOGHEADER => 'domainid  hh:mm:ss  opid'  *
*                                                                     *
***********************************************************************
*
*    Blank out log buffer
*
         BAL   R14,CLRBFR         Clear the buffer
*
*    Get DATE and TIME
*
         LA    R8,LOGDOMID        ADDRESS LOG DATIM
         DSIDATIM AREA=(R8)       GET DATE AND TIME
*
*    Overwrite date portion of DATIM with Domain Id
*
         MVC   0(L'MVTCURAN,R8),MVTCURAN
*
*    Get opid
*
         LA    R9,TVBOPID         ADDRESS TASKID (OPID)
         MVC   OPID(L'OPID),0(R9) Copy opid into log
*****************************************************************
*
*    Get text to be logged from command buffer.
*
*****************************************************************
         L     R6,CWBBUF                 ADDRESS COMMAND BUFFER
         LH    R11,HDRMLENG-BUFHDR(R6)   R11 has total text length
         AH    R6,HDRTDISP-BUFHDR(R6)    R6 points to text
*
*  Get length of the command
*
         L     R7,CWBPDB                 ADDRESS PDB
         LA    R8,PDBTABLE-DSIPDB(R7)    ADDRESS PDB ENTRIES
         SLR   R9,R9                     CLEAR WORK REG
         IC    R9,PDBLENG-PDBENTRY(R8)   LENGTH OF COMMAND
         LA    R9,1(R9)                  Add blank after command
*
*  Check for no text
*
         SR    R11,R9                    SUBTRACT LENGTH OF COMMAND
         BNP   LOGIT                     JUST LOG HEADER IF NO TEXT
         AR    R6,R9                          Add length to text pointer
*
*  Log text 1 buffer at a time until all text is logged
*
CHKLEN   EQU   *
         LA    R10,MAXTXTLN                   GET MAX TEXT WE CAN LOG
         CR    R11,R10                        Is text left > max
         BNL   COPYTEXT                       No - Then log max
         LR    R10,R11                        Yes - Then log whats left
COPYTEXT EQU   *
         SR    R11,R10                        Dec the total length
         LR    R9,R6                          Address text to be copied
         AR    R6,R10                         Update text pointer
         LA    R8,LOGTEXT                     Address text area
         BCTR  R10,0                          Decrement for execute
         EX    R10,COPYMSG                    Copy text into buffer
LOGIT    EQU   *
*
         L     R7,CWBSWB                      Address the SWB
         LA    R8,LOGBFR                      Address log buffer
         DSIWLS SWB=(R7),SAMREC=(R8),SAMLEN=SQLEN,SAMTASK=SQTASK
*                                             R15 will have RC from DSIWLS
         LTR   R15,R15                        Was logging successful?
         BNZ   SLOGERR                        No - inform the issuer
*                                             Yes - check for more text
         LTR   R11,R11                        More text?
         BNP   EXIT                           No - then exit
         BAL   R14,CLRBFR                     Yes - clear buffer
         B     CHKLEN                         Branch back to CHkLEN
SLOGERR  EQU   *
******
*   Get the Error message
******
         LA    R6,MSGBFR                  ADDRESS MESSAGE BUFFER
         LR    R8,R6                      COPY BUFFER ADDRESS
         AH    R8,HDRTDISP-BUFHDR(R6)     POINT TO TEXT AREA
         LA    R9,BADRC                   ADDRESS BAD RETURN CODE MESSAGE
         LA    R10,L'BADRC                LENGTH OF MESSAGE
         LR    R11,R10                    SAVE THIS LENGTH
         BCTR  R10,0                      DECREMENT FOR EXECUTE
         EX    R10,COPYMSG                COPY MESSAGE TEXT INTO BUFFER
******
*   Make the RC printable
******
DECCONV  EQU   *
         CVD   R15,DBLEWORD                 Convert value to dec
         MVC   MASK,=X'402020202120'        Get the mask
         ED    MASK,DBLEWORD+5              Edit the converted value
         ALR   R8,R11                       Point to end of bad rc msg
         MVC   0(2,R8),MASK+4               Move to converted rc to output
         A     R11,=F'2'                    Calculate total length
         STH   R11,HDRMLENG-BUFHDR(R6)      Store length in header
         BAL   R14,SENDMSG                  Send the message
*                                           If message fails, just
*                                           exit with bad return code
         LA    R15,ERRCODE
         B     EXIT
************************************************************************
* Send a message                                                       *
************************************************************************
SENDMSG  DS    0H
         ST    R14,SAVERET
         LA    R6,MSGBFR
         L     R7,CWBSWB
         CLI   CBHTYPE-DSICBH(R3),CBHOST CHECK FOR OST
         BE    USEPSS               USE DSIPSS IF OST
         CLI   CBHTYPE-DSICBH(R3),CBHNNT CHECK FOR NNT
         BE    USEPSS              USE DSIPSS IF NNT
         DSIMQS BFR=(R6),SWB=(R7),AUTHRCV=YES
         B     EXITSUB
USEPSS   EQU   *
         DSIPSS BFR=(R6),SWB=(R7),TYPE=OUTPUT
EXITSUB  EQU   *
         L     R14,SAVERET
         BR    R14
***********************************************************************
* MODULE EXIT                                                         *
***********************************************************************
EXIT     EQU   *
         L     R13,4(R13)
         L     R14,12(R13)
         LM    R0,R12,20(R13)
         BR    R14
***********************************************************************
* Routine to blank out buffer                                         *
***********************************************************************
*
CLRBFR   EQU   *
         ST    R14,SAVERET         Save return address
         LA    R9,LOGBFR           POINT TO LOG BUFFER
         LA    R8,1(R9)            AND THE NEXT POSITION
         MVI   0(R9),X'40'         PUT BLANK IN FIRST BYTE
         LA    R10,LOGBFRLN        GET LENGTH OF LOG BUFFER
         BCTR  R10,0               DECREMENT FOR EXECUTE
         EX    R10,COPYMSG         PROPAGATE BLANKS FOR ENTIRE BFR
         L     R14,SAVERET         RESTORE RETURN ADDRESS
         BR    R14                 RETURN
***********************************************************************
***********************************************************************
* CONSTANTS, EQUATES, AUTODATA SET-UP, ETC.                           *
***********************************************************************
         DS    0H
COPYMSG  MVC   0(0,R8),0(R9)
**********************************************************************
* ENSURE THAT NO TEXT LINE EXTENDS BEYOND THE TEXT AREA ALLOCATED    *
* FOR THE MESSAGE BUFFER IN AUTODATA.                                *
**********************************************************************
SQTASK   DC    CL8'SQLOGTSK'
SQLEN    DC    F'100'
INVTASK  DC    CL32'COMMAND INVALID FOR THIS TASK - '
BADRC    DC    CL57'SLOG000  ERROR ENCOUNTERED ISSUING DSIWLS, RETURN CX
               ODE = '
ERRCODE  EQU   4
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
DSICWB   DSECT
         ORG   CWBADATD
AUTOST   EQU   *                    START OF AUTODATA
```

```
SAVERET  DS    F                       SAVE SPACE FOR RETURN ADDRESS
DBLEWORD DS    D
MASK     DS    CL6
MSGBFR   DS    CL(BUFHDRND-BUFHDR+100)   ***NOTE: This buffer header
         ORG   MSGBFR                      is not needed for sequential
MSGHDR   DS    CL(BUFHDRND-BUFHDR)          logging it is included for
LOGBFR   EQU   *                            sending messages
         DS    CL1
LOGDOMID DS    CL8
         DS    CL1
OPID     DS    CL8
         DS    CL1
LOGHDRLN EQU   *-LOGBFR
LOGTEXT  EQU   *
TXTAREA  DS    CL81
MAXTXTLN EQU   *-LOGTEXT
LOGBFRLN EQU   *-LOGBFR
AUTOEND  EQU   *                       END OF AUTODATA
*                                      THE FOLLOWING IS TO HELP WARN
*                                      ABOUT INSUFFICIENT AUTODATA
AUTOREM  EQU   (L'CWBADATD-(AUTOEND-AUTOST))
         END   ASEQLOG
```

Glossary, Bibliography, and Index

Glossary

This glossary defines important NCP, NetView, NetView/PC, SSP, and VTAM abbreviations and terms. It includes information from the *IBM Dictionary of Computing*, SC20-1699. Definitions from the *American National Dictionary for Information Processing* are identified by an asterisk (*). Definitions from draft proposals and working papers under development by the International Standards Organization, Technical Committee 97, Subcommittee 1 are identified by the symbol **(TC97)**. Definitions from the *CCITT Sixth Plenary Assembly Orange Book, Terms and Definitions* and working documents published by the Consultative Committee on International Telegraph and Telephone of the International Telecommunication Union, Geneva, 1980 are preceded by the symbol **(CCITT/ITU)**. Definitions from published sections of the *ISO Vocabulary of Data Processing*, developed by the International Standards Organization, Technical Committee 97, Subcommittee 1 and from published sections of the *ISO Vocabulary of Office Machines*, developed by subcommittees of ISO Technical Committee 95, are preceded by the symbol **(ISO)**.

For abbreviations, the definition usually consists only of the words represented by the letters; for complete definitions, see the entries for the words.

Reference Words Used in the Entries

The following reference words are used in this glossary:

Deprecated term for. Indicates that the term should not be used. It refers to a preferred term, which is defined.

Synonymous with. Appears in the commentary of a preferred term and identifies less desirable or less specific terms that have the same meaning.

Synonym for. Appears in the commentary of a less desirable or less specific term and identifies the preferred term that has the same meaning.

Contrast with. Refers to a term that has an opposed or substantively different meaning.

See. Refers to multiple-word terms that have the same last word.

See also. Refers to related terms that have similar (but not synonymous) meanings.

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/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 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*.

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.

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.

allocate. A logical unit (LU) 6.2 application program interface (API) verb used to assign a session to a conversation for the conversation's use. Contrast with *deallocate*.

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.

authorized receiver. In the NetView program, an authorized operator who receives all the unsolicited and authorized-receiver messages not assigned to a specific operator.

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.

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*.

begin bracket. In SNA, the value (binary 1) of the begin-bracket indicator in the request header (RH) of the first request in the first chain of a bracket; the value denotes the start of a bracket. Contrast with *end bracket*. See also *bracket*.

bidder. In SNA, the LU-LU half-session defined at session activation as having to request and receive permission from the other LU-LU half-session to begin a bracket. Contrast with *first speaker*. See also *bracket protocol* and *contention*.

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* (including illustration). 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.

bracket. In SNA, one or more chains of request units (RUs) and their responses that are exchanged between the two LU-LU half-sessions and that represent a transaction between them. A bracket must be completed before another bracket can be started. Examples of brackets are data base inquiries/replies, update transactions, and remote job entry output sequences to work stations. See also *begin bracket* and *end bracket*.

bracket protocol. In SNA, a data flow control protocol in which exchanges between the two LU-LU half-sessions are achieved through the use of brackets, with one LU designated at session activation as the first speaker and the other as the bidder. The bracket protocol involves bracket initiation and termination rules. See also *bidder* and *first speaker*.

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

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

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.

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

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

CCP. Configuration control program facility.

CDRM. Cross-domain resource manager.

CDRSC. Cross-domain resource.

chain. (1) A group of logically linked records. (2) See *RU chain*.

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-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*.

character-coded. Synonym for *unformatted*.

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.

cleanup. A network services request, sent by a system services control unit (SSCP) to a logical unit (LU), that causes a particular LU-LU session with that LU to be ended immediately and without the participation of either the other LU or its SSCP.

CLIST. Command list.

CMS. Conversational Monitor System.

CNM. Communication network management.

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 processor written in a high-level language (HLL) or a command list. See also *command list* and *command processor*.

command processor. (1) A program that performs an operation specified by a command. (2) In the NetView program, 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 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. 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.

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 control program (CCP) facility. An SSP interactive application program facility by which configuration definitions for the IBM 3710 Network Controller can be created, modified, and maintained.

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*.

connection. Synonym for *physical connection*.

contention. A situation in which two logical units (LUs) that are connected by an LU 6.2 session both attempt to allocate the session for a conversation at the same time. The control operator assigns "winner" and "loser" status to the LUs so that processing may continue on an orderly basis. The contention loser requests permission from the contention winner to allocate a conversation on the session, and the contention winner either grants or rejects the request. See also *bidder*.

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.

converted command. An intermediate form of a character-coded command produced by VTAM through use of an unformatted system services definition table.

The format of a converted command is fixed; the unformatted system services definition table must be constructed in such a manner that the character-coded command (as entered by a logical unit) is converted into the predefined, converted command format. See also *unformatted.*

COS. Class of service.

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.

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

data flow control (DFC) layer. In SNA, the layer within a half-session that (1) controls whether the half-session can send, receive, or concurrently send and receive request units (RUs); (2) groups related RUs into RU chains; (3) delimits transactions via the bracket protocol; (4) controls the interlocking of requests and responses in accordance with control modes specified at session activation; (5) generates sequence numbers; and (6) correlates requests and responses.

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 request block (DSRB). The control block in the NetView program that contains information that a data services command processor (DSCP) needs to communicate with the data services task (DST).

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.

DBCS. Double-byte character set.

ddname. Data definition name.

deallocate. A logical unit (LU) 6.2 application program interface (API) verb that terminates a conversation, thereby freeing the session for a future conversation. Contrast with *allocate.*

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 18, Figure 19, and Figure 20 on page 252. See also *macroinstruction.*

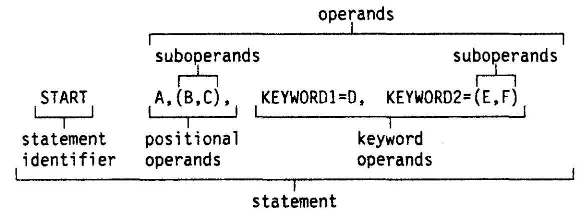


Figure 18. Example of a Language Statement

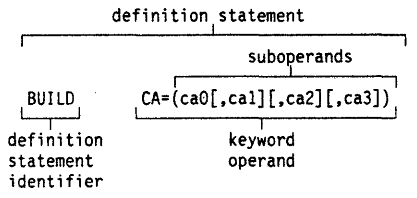
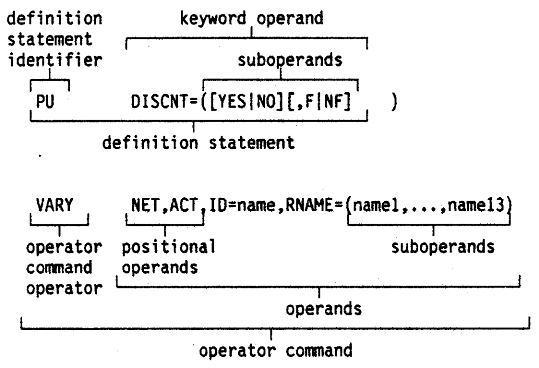


Figure 19. NCP Examples


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 20. VTAM Examples

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

direct call. (ISO) A facility that permits calling without requiring the user to provide address selection signals; the network interprets the call request signal as an instruction to establish a connection to one or more predetermined data stations.

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*.

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 Japanese, 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*.

drop. In the IBM Token-Ring Network, a cable that leads from a faceplate to the to the distribution panel in a wiring closet. When the IBM Cabling System is used with the IBM Token-Ring Network, a drop may form part of a lobe.

DSCP. Data services command processor.

DST. Data services task.

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

ECB. Event control block.

echo. The return of characters to the originating SS device to verify that a message was sent correctly.

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 bracket. In SNA, the value (binary 1) of the end bracket indicator in the request header (RH) of the first request of the last chain of a bracket; the value denotes the end of the bracket. Contrast with *begin bracket*. See also *bracket*.

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*.

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.

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 list (EXLST). In VSAM and VTAM, a control block that contains the addresses of routines that receive control when specified events occur during execution; for example, routines that handle session-establishment request processing or I/O errors.

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*.

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

field-formatted. Pertaining to a request or response that is encoded into fields, each having a specified format such as binary codes, bit-significant flags, and symbolic names. Contrast with *character-coded*.

first speaker. In SNA, the LU-LU half-session defined at session activation as: (1) able to begin a bracket without requesting permission from the other LU-LU half-session to do so, and (2) winning contention if both half-sessions attempt to begin a bracket simultaneously. Contrast with *bidder*. See also *bracket protocol*.

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. An entry point that provides centralized management and control for other entry points for one or more network management categories.

formatted system services. A portion of VTAM that provides certain system services as a result of receiving a field-formatted command, such as an Initiate or Terminate command. Contrast with *unformatted system services (USS)*. See also *field-formatted*.

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*.

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.

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. A printed copy of machine output in a visually readable form; for example, printed reports, listings, documents, summaries, or network logs.

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*.

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.

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*.

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*.

initiate. A network services request sent from a logical unit (LU) to a system services control point (SSCP) requesting that an LU-LU session be established.

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.

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.

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

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*.

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 21 on page 256.

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

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.

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.

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 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.

log on. To initiate a session.

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 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.

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-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.

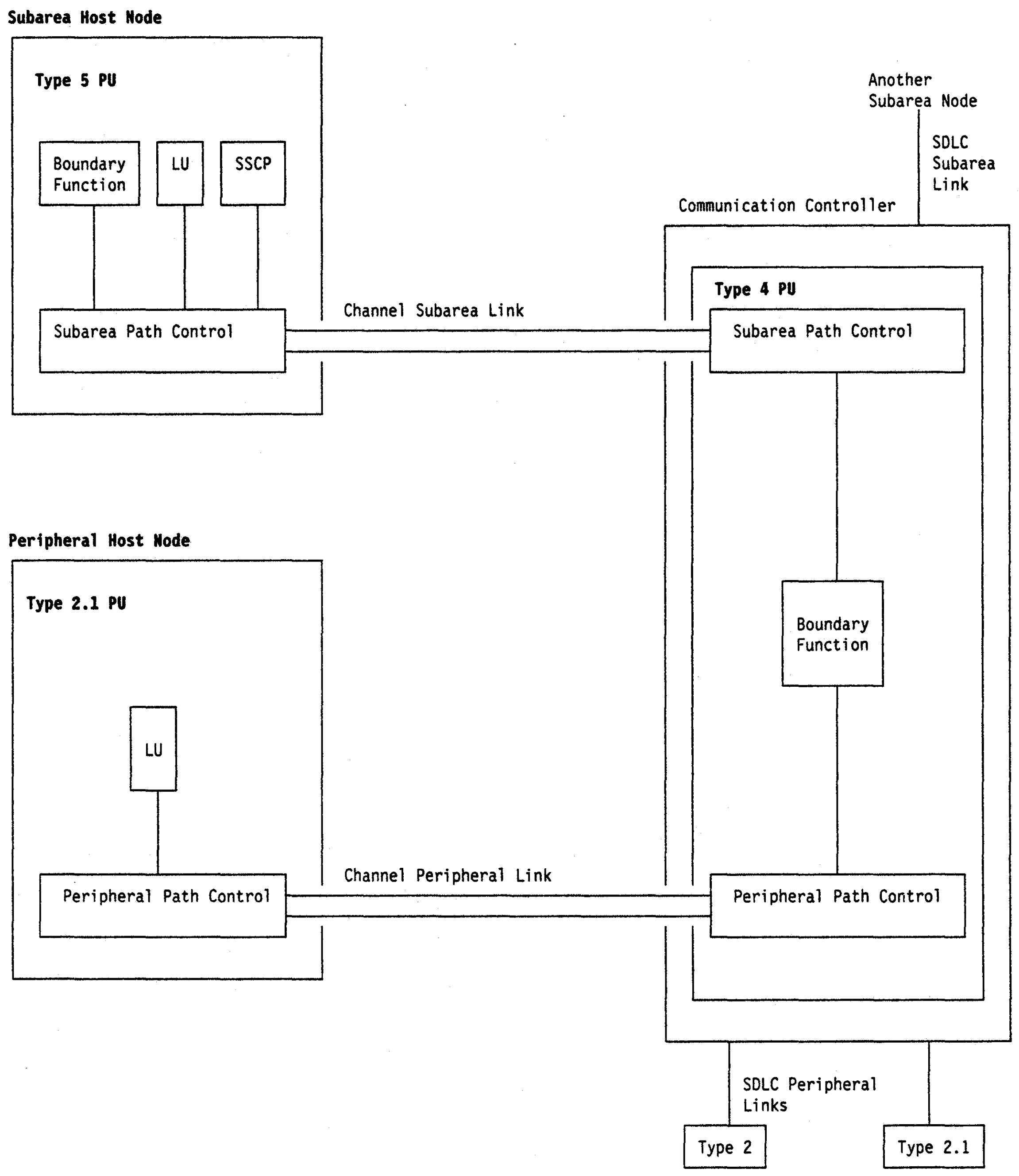


Figure 21. Links and Path Controls

LU-LU session type. A deprecated term for *LU type*.

LU 6.2. Logical unit 6.2.

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*.

Medium Access Control (MAC). The sublayer of DLC that supports medium-dependent functions and uses the services of the physical layer to provide services to Logical Link Control (LLC). The MAC sublayer includes the medium access port.

medium access control (MAC) procedure. (TC97) In a local area network, the part of the protocol that governs access to the transmission medium independently of the physical characteristics of the medium, but takes into account the topological aspects of the network, in order to enable the exchange of data between data stations.

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.

migration. Installing a new version or release of a program when an earlier version or release is already in place.

minor node. In VTAM, a uniquely-defined resource within a major node. See *node* and *major node*.

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-domain network. In SNA, a network with more than one system services control point (SSCP). Contrast with *single-domain network*.

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.

MVS. Multiple Virtual Storage operating system.

MVS/XA. Multiple Virtual Storage for Extended Architecture operating system.

NAU. Network addressable unit.

NC. Network control.

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-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, subarea network, type 2.1 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 (NC). In SNA, an RU category used for requests and responses exchanged for such purposes as activating and deactivating explicit and virtual routes and sending load modules to adjacent peripheral nodes. See also *data flow control layer* and *session control*.

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 control program. A program, generated by the user from a library of IBM-supplied modules, that controls the operation of a communication controller.

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 product support (NPS). The function of the NetView program that provides operations control for the IBM 3710 Network Controller, 5860 family of modems, and the NCP; and configuration of 3710s and the 5860 family of modems. NPS provides operator commands to run diagnostics for link problem determination and to change product operating parameters.

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*.

network services (NS) header. In SNA, a 3-byte field in a function management data (FMD) request/response unit (RU) flowing in an SSCP-LU, SSCP-PU, or SSCP-SSCP session. The network services header is used primarily to identify the network services category of the request unit (RU) (for example, configuration services, session services) and the particular request code within a category.

NMVT. Network management vector transport.

NNT. NetView-NetView task.

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* (including illustration). (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*.

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*.

NPDA. Network Problem Determination Application.

NPS. Network product support.

NS. Network services.

OCCF. Operator Communication Control Facility.

online. Stored in a computer and accessible from a terminal.

open. (1) In the IBM Token-Ring Network, to make an adapter ready for use. (2) A break in an electrical circuit.

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 Communication Control Facility (OCCF). A licensed program that allows communication with and the operation of remote MVS or VSE systems.

operator profile. In the NetView program, the resources and activities a network operator has control over. The statements defining these resources and activities are stored in a file that is activated when the operator logs on.

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.

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 group. In SNA, (1) The path information units (PIUs) that can be transmitted on a virtual route before a virtual-route pacing response is received, indicating that the virtual route receiver is ready to receive more PIUs on the route. Synonymous with *window*. (2) The requests that can be transmitted on the normal flow in one direction on a session before a session-level pacing response is received, indicating that the receiver is ready to accept the next group of requests.

pacing group size. In SNA, (1) The number of path information units (PIUs) in a virtual route pacing group. The pacing group size varies according to traffic congestion along the virtual route. Synonymous with *window size*. (2) The number of requests in a session-level pacing group.

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.

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 error. Synonym for *temporary error*.

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.

permanent error. A resource error that cannot be resolved by error recovery programs. Contrast with *temporary error*.

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.

presentation services command processor (PSCP). In the NetView program, a facility that processes requests from a user terminal and formats displays to be presented at the user terminal.

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.

profile. In the Conversational Monitor System (CMS) or the group control system (GCS), the characteristics defined by a PROFILE EXEC file that executes automatically after the system is loaded into a virtual machine. See also *operator profile*.

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.

protection key. An indicator that appears in the current program status word whenever an associated task has control of the system. This indicator must match the storage keys of all main storage locks that the task is to use.

protocol. (1) (CCITT/ITU) A specification for the format and relative timing of information exchanged between communicating parties. (2) (TC97) The set of rules governing the operation of functional units of a communication system that must be followed if communication is to be achieved. (3) In SNA, the meanings of, and the sequencing rules for, requests and responses used for managing the network, transferring data, and synchronizing the states of network components. See also *bracket protocol*. Synonymous with *line control discipline* and *line discipline*. See also *link protocol*.

PSCP. Presentation services command processor.

PU. Physical unit.

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*.

PU-PU flow. In SNA, the exchange between physical units (PUs) of network control requests and responses.

receive pacing. In SNA, the pacing of message units that the component is receiving. See also *send pacing*.

RECFMS. Record formatted maintenance statistics.

RECMS. Record maintenance statistics.

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.

record formatted maintenance statistics (RECFMS). A statistical record built by an SNA controller and usually solicited by the host.

record maintenance statistics (RECMS). An SNA error event record built from an NCP or line error and sent unsolicited to the host.

reentrant. The attribute of a program or routine that allows the same copy of the program or routine to be used concurrently by two or more tasks. For example, the 3710 Network Controller routines may be reentrant.

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.

remove. In the IBM Token-Ring Network, to take an attaching device off the ring.

REQMS. Request for maintenance statistics.

request for maintenance statistics (REQMS). A host solicitation to an SNA controller for a statistical data record.

request header (RH). In SNA, control information preceding a request unit (RU). See also *request/response header (RH)*.

request parameter list (RPL). In VTAM, a control block that contains the parameters necessary for processing a request for data transfer, for establishing or terminating a session, or for some other operation.

request unit (RU). In SNA, a message unit that contains control information, end-user data, or both.

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.

request/response unit (RU). In SNA, a generic term for a request unit or a response unit. See also *request unit (RU)* and *response unit*.

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 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*.

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 unit (RU). In SNA, a message unit that acknowledges a request unit; it may contain prefix information received in a request unit. If positive, the response unit may contain additional information (such as session parameters in response to Bind Session), or if negative, contains sense data defining the exception condition.

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.

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)*.

routing. The assignment of the path by which a message will reach its destination.

RPL. Request parameter list.

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*.

RU. Request/response unit.

RU chain. In SNA, a set of related request/response units (RUs) that are consecutively transmitted on a particular normal or expedited data flow. The request RU chain is the unit of recovery: if one of the RUs in the chain cannot be processed, the entire chain is discarded. Each RU belongs to only one chain, which has a beginning and an end indicated by means of control bits in request/response headers within the RU chain. Each RU can be designated as first-in-chain (FIC), last-in-chain (LIC), middle-in-chain (MIC), or only-in-chain (OIC). Response units and expedited-flow request units are always sent as only-in-chain.

same-domain. Refers to communication between entities in the same SNA domain. Contrast with *cross-domain*. See also *single-domain network*.

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*.

sequence number. A number assigned to a particular frame or packet to control the transmission flow and receipt of data.

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 reminder (SR). In the NetView/PC program, a notification set by the operator that is displayed on a panel and logs a specified message.

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 control (SC). In SNA, (1) One of the components of transmission control. Session control is used to purge data flowing in a session after an unrecoverable error occurs, to resynchronize the data flow after such an error, and to perform cryptographic verification. (2) A request unit (RU) category used for requests and responses exchanged between the session control components of a session and for session activation and deactivation requests and responses.

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*.

session monitor. The component of the NetView program that collects and correlates session-related data and provides online access to this information.

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*.

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.

solicited message. A response from VTAM to a command entered by a program operator. Contrast with *unsolicited message*.

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.

SR. Service reminder.

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.

start option. In VTAM, a user-specified or IBM-supplied option that determines certain conditions that are to exist during the time a VTAM system is operating. Start options can be predefined or specified when VTAM is started.

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.

status monitor. A component of the NetView program that collects and summarizes information on the status of resources defined in a VTAM domain.

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 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.

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.

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.

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.

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.

TAF. Terminal access facility.

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*.

temporary error. A resource failure that can be resolved by error recovery programs. Synonymous with *performance error*. Contrast with *permanent error*.

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.

terminal access facility (TAF). In the NetView program, a facility that allows a network operator to control a number of subsystems. In a full-screen or operator control session, operators can control any combination of such subsystems simultaneously.

TERMINATE. In SNA, a request unit that is sent by a logical unit (LU) to its system services control point (SSCP) to cause the SSCP to start a procedure to end one or more designated LU-LU sessions.

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.

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.

tutorial. Online information presented in a teaching format.

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.

unformatted. In VTAM, pertaining to commands (such as LOGON or LOGOFF) entered by an end user and sent by a logical unit in character form. The character-coded command must be in the syntax defined in the user's unformatted system services definition table. Synonymous with *character-coded.* Contrast with *field-formatted.*

unformatted system services (USS). In SNA products, a system services control point (SSCP) facility that translates a character-coded request, such as a logon or logoff request into a field-formatted request for processing by formatted system services and translates field-formatted replies and responses into character-coded requests for processing by a logical unit. Contrast with *formatted system services.* See also *converted command.*

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.

unsolicited message. A message, from VTAM to a program operator, that is unrelated to any command entered by the program operator. Contrast with *solicited message.*

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-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.*

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).

USS. Unformatted system services.

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.

verb. (1) In SNA, the general name for a transaction program's request for communication services. (2) In VTAM, a programming language element in the logical unit (LU) 6.2 application program interface (API) that causes an LU 6.2 function to be performed.

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 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. (ISO) The notion of storage space that may be regarded as addressable main storage by the user of a computer system in which virtual addresses are mapped into real addresses. The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of auxiliary storage available, not by the actual number of main storage locations.

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 operating system. Its full name is Virtual Machine/System Product. Synonymous with *VM/SP*.

VM SNA console support (VSCS). A VTAM component for the VM environment that provides Systems Network Architecture (SNA) support. It allows SNA terminals to be virtual machine consoles.

VM/SP. Virtual Machine/System Product operating system. Synonym for *VM.*

VR. Virtual route.

VSAM. Virtual Storage Access Method.

VSCS. VM SNA console support.

VSE. Virtual Storage Extended operating system. Synonymous with *VSE/AF.*

VSE/AF. Virtual Storage Extended/Advanced Function operating system. Synonym for *VSE*.

VSE/OCCF. Virtual Storage Extended/Operator Communication Control Facility.

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 application program. A program that has opened an ACB to identify itself to VTAM and can now issue VTAM macroinstructions.

VTAM operator command. A command used to monitor or control a VTAM domain. See also *definition statement.*

window. (1) In SNA, synonym for *pacing group*. (2) On a visual display terminal, a small amount of information in a framed-in area on a panel that overlays part of the panel. (3) In data communication, the number of data packets a data terminal equipment (DTE) or data circuit-terminating equipment (DCE) can send across a logical channel before waiting for authorization to send another data packet. The window is the main mechanism of pacing, or flow control, of packets.

window size. (1) The specified number of frames of information that can be sent before receiving an acknowledgment response. (2) In SNA, synonym for *pacing group size.*

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)[6] 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.

[6] This guide will be available in July 1989.

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)[7]

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)

[7] This book will be available by December 1989.

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

VM/SP System Product Interpreter Reference or *TSO/E REXX Reference* (SC28-1883) (referred to in this book as *REXX Reference*.)

IBM 3600/4700 Threshold Analysis and Remote Access Feature: General Information (GC34-2055)

IBM 3600/4700 Threshold Analysis and Remote Access Feature: User's Guide (SC34-2056)

IBM 3600/4700 Threshold Analysis and Remote Access Feature: Installation and Customization (SC34-2041)

Index

Reader's Comment Form

NetView™
Customization: Using Assembler
Release 3

Publication No. SC31-6078-1

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note: Copies of IBM Publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Possible topics for comment are: clarity, accuracy, completeness, organization, coding, retrieval, and legibility.

Comments: ______________________________

____

____

____

____

____

____

What is your occupation? ______________________

If you wish a reply, give your name, company, mailing address, and date:

____

____

____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

SC31-6078-1

Reader's Comment Form

Fold and tape **Please Do Not Staple** **Fold and tape**

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Dept. E15
P.O. Box 12195
Research Triangle Park, N.C. 27709-9990

Fold and tape **Please Do Not Staple** **Fold and tape**

IBM ®