H300e NetEx/IP™

for Unisys OS2000/Dorado Systems

Release 7.0.5

Software Reference Manual

MAN-REF-H300e-7.0.5

Revision Record

Revision	**Description**
7.0.4 (**2/2017**)	Manual released.
7.0.5 (4/2018)	Minor updates for Release 7.0.5

You may submit written comments to:

Network Executive Software, Inc.
Publications Department
6450 Wedgwood Road N, Suite 103
Maple Grove, MN 55311
USA

Comments may also be submitted over the Internet by addressing e-mail to:

support@netex.com

or, by visiting our web site at:

http://www.netex.com

Always include the complete title of the document with your comments.

Preface

This manual describes the extended mode NetEx/IP™ software for Unisys OS2200 on Dorado platforms.

"Chapter 1: Introduction", "Chapter 2: NetEx/IP and the ISO Model", "Chapter 3: NetEx/IP Session Services", and "Chapter 4: NetEx Request Block" are intended for all readers. "Chapter 5: C High Level Interface" describes the library of subroutines that are called by the C high-level language programs. "Appendix A: NRB Error Codes" includes a list and description of the error messages and codes issued by NetEx/IP.

Readers are not expected to be familiar with NetEx/IP before using this manual. However, an understanding of programming and using the host operating system is required.

Reference Material

The following manuals contain related information.

Number	Title and Description
MAN-CNET-CONF-MGR	*"C" Configuration Manager and NETEX Alternate Path Retry (APR) User Guide*

Notice to the Reader

Corporation Trademarks and Products

Network Executive Software	**NetEx, NetEx/IP, BFX, PFX, USER-Access, eFT**
International Business Machines	**IBM**
Unisys Corporations	**OS2200, Dorado**

These references are made for informational purposes only.

The diagnostic tools and programs described in this manual are **not** part of the products described.

Notice to the Customer

Installation information contained in this document is intended for use by experienced System Programmers.

Document Conventions

The following notational conventions are used in this document.

Format	Description
`displayed information`	Information displayed on a CRT (or printed) is shown in `this font`.
`user entry`	*`This font`* is used to indicate the information to be entered by the user.
UPPERCASE	The exact form of a keyword that is not case-sensitive or is issued in uppercase.
MIXedcase	The exact form of a keyword that is not case-sensitive or is issued in uppercase, with the minimum spelling shown in uppercase.
bold	The exact form of a keyword that is case-sensitive and all or part of it must be issued in lowercase.
lowercase	A user-supplied name or string.
value	Underlined parameters or options are defaults.
<label>	The label of a key appearing on a keyboard. If "label" is in uppercase, it matches the label on the key (for example: <ENTER>). If "label" is in lowercase, it describes the label on the key (for example: <up-arrow>).
<key1><key2>	Two keys to be pressed simultaneously.
No delimiter	Required keyword/parameter.

Glossary

asynchronous: A class of data transmission service whereby all requests for service contend for a pool of dynamically allocated ring bandwidth and response time.

ASCII: Acronym for American National Standard Code for Information Interchange.

buffer: A contiguous block of memory allocated for temporary storage of information in performing I/O operations. Data is saved in a predetermined format. Data may be written into or read from the buffers.

code conversion: An optional feature in NetEx that dynamically converts the user data from one character set to another (for example, ASCII, EBCDIC, et cetera).

Configuration Manager: A utility that parses a text NCT file into a PAM file.

header: A collection of control information transmitted at the beginning of a message, segment, datagram, packet, or block of data.

host: A data processing system that is connected to the network and with which devices on the network communicate. In the context of Internet Protocol (IP), a host is any addressable node on the network; an IP router has more than one host address.

Internet Protocol (IP): A protocol suite operating within the Internet as defined by the *Requests For Comment* (RFC). This may also refer to the network layer (level 3) of this protocol stack (the layer concerned with routing datagrams from network to network).

ISO: Acronym for International Standards Organization.

Network Configuration Table (NCT): An internal data structure that is used by the NetEx configuration manager program to store all the information describing the network.

NETwork EXecutive (NetEx): A family of software designed to enable two or more application programs on heterogeneous host systems to communicate. NetEx is tailored to each supported operating system, but can communicate with any other supported NetEx, regardless of operating system.

NetEx can reside on the host or another host with TNP.

NetEx is a registered trademark of Network Executive Software.

Open Systems Interconnection (OSI): A seven-layer protocol stack defining a model for communications among components (computers, devices, people, etcetera) of a distributed network. OSI was defined by the ISO.

path: A route that can reach a specific host or group of devices.

TCP/IP: An acronym for Transmission Control Protocol/Internet Protocol. These communication protocols provide the mechanism for inter-network communications, especially on the Internet. The protocols are hardware-independent. They are described and updated through *Requests For Comment* (RFC). IP corresponds to the OSI network layer 3, TCP to layers 4 and 5.

Contents

Figures

Tables

Chapter 1: Introduction

Network Executive Software's NetEx/IP™ allows two or more application programs (which may be on different host computers) to communicate with each other at multi-megabit speeds. The NetEx/IP family of software consists of different versions of NetEx/IP for use with different operating systems, such as the versions for use with the various zOS, OS2200, UNIX and Windows operating system hosts. All of these versions provide a common high-level interface to simplify programming requirements. NetEx/IP utility programs are also available, such as the Bulk File Transfer (BFX™), Print File Transfer (PFX™), and USER-Access® utilities.

NetEx/IP Characteristics

NetEx/IP centralizes network considerations for IP networks, into a single piece of software. The following sections describe the characteristics of the NetEx/IP software.

- External interface
- Internal interaction
- NetEx/IP connections
- Design flow efficiency and flexibility
- Block segmenting
- Alternate Path Retry
- Basic I/O flow
- Remote operator interface

External Interface

The NetEx/IP external interface for the application programmer is common for all versions of NetEx/IP. NetEx/IP provides requests for use in the programs that call NetEx/IP. These calling programs may be written in C or other high-level languages. NetEx/IP programs written in high-level languages may be transported from one host to another, with some changes to account for different word sizes and other machine architecture variations.

NetEx/IP also provides an operator interface that monitors and controls certain NetEx/IP functions.

Internal Interaction

The internal operation of all supported versions of NetEx/IP are consistent and allows the different versions to interact freely. Thus, any program using NetEx/IP may communicate with any other program on the network that is also using NetEx/IP. When a NetEx application initiates a session (offer/connect), NetEx/IP utilizes the UDP system services of the IP stack to establish the NetEx session and transfer data between NetEx/IP nodes. The default port used in the NetEx/IP network is 6950. This port number can be changed, but must be the same for all nodes in the NetEx/IP network.

Note: NetEx/IP's UDP port (6950 by default) must be allowed through firewalls between NetEx/IP nodes.

To facilitate communication between hosts of different manufacture, NetEx/IP supports code conversion. NetEx/IP software can perform code conversion.

NetEx/IP Connections

To communicate using NetEx/IP, two calling programs first form a connection using a handshake protocol. NetEx/IP then allows this pair of programs to communicate.

NetEx/IP can establish multiple connections at one time, and can allow one program to have multiple connections simultaneously.

NetEx/IP also supports communications within a single host. A calling program may connect to another calling program in the same host and exchange information just as if network communications were taking place.

Design Efficiency and Flexibility

The NetEx/IP design enables many applications on the same processor to share the use of the network facility. Programs calling NetEx/IP can be written without regard to the other programs calling NetEx/IP or other Network Executive Software device drivers.

Once NetEx/IP accepts data from the caller, NetEx/IP must deliver the data to its destination. The NetEx/IP subsystem on each host handles flow control, error recovery, and any other special considerations such as latency.

NetEx/IP optimizes data transfer throughput using a high degree of parallelism. That is, under normal circumstances, simultaneous adapter I/O, NetEx/IP buffer management, and user file I/O all take place concurrently. This means that the effective data transfer rate is as fast as possible (in the multi-megabit range).

Block Segmenting

NetEx/IP products provide block segmenting at the transport layer. NetEx/IP divides data into segments of a specified size for transmission across the network and reassembles the segments on the remote host before delivering the data to the session layer calling program on the remote NetEx/IP. This segmenting is transparent to the session user but provides control of the transmitted block segment size. This is especially useful for networks with long latency.

Alternate Path Retry

Alternate Path Retry (APR) provides the capability for connections to automatically reroute on different network paths when a failure on a path is detected. This rerouting takes place with no loss of data. Alternate path retry is provided as part of the type 2 protocol supplied with current NetEx/IP versions. For more information on APR, refer to the *"C" Configuration Manager and NETEX Alternate Path Retry (APR) User Guide*.

Remote Operator Interface

This version of NetEx/IP provides a remote operator interface that allows users to issue NetEx/IP operator commands to other defined NetEx/IP hosts on the network. Other users may also be the remote operator for this NetEx/IP. See "Enabling the Remote Operator Service" for more information. Security features are provided.

New Features

NetEx/IP release 7.0.4 provides support for standard IP networks as well as introducing NetEx/IP protocol extensions (referred to as Type 4 protocol), that provide the ability for NetEx/IP to dynamically maximize the network performance, based on factors such as available bandwidth, distance, and workload on the network. *(This protocol type is not yet supported on all platforms, including Unisys.)*

Basic I/O Flow

Figure 1 shows the basic I/O flow between two programs using host based NetEx/IP. The calling program communicates with NetEx/IP through the NetEx/IP user interface. NetEx/IP then uses the available network hardware to communicate with the calling program on the other processor.

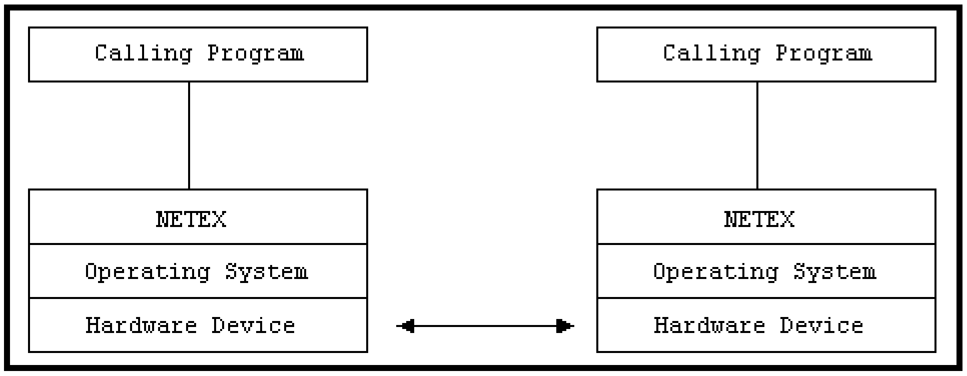


Figure 1. Basic I/O Flow

Host Based NetEx/IP

Host based NetEx exists on the machine as a subsystem (a separate program residing in a machine that all other users in the machine can call on to perform services). User tasks produce a NetEx request that is delivered to the independent NetEx program using an inter-task communications facility provided by the host operating system. Data is moved so it is present in the NetEx program and the I/O is performed in the NetEx program.

Host based NetEx provides an administrative capability to the system programmers and system managers. Since all I/O is performed by the NetEx program, no data can be introduced on the network without first being checked by NetEx.

NetEx/IP via TNP

The NetEx/IP program may reside in another host running TNP. Only the Hxx7IP NetEx Requester user interface program resides on the local host. In this implementation, H267IP the OpenVMS TCP/IP product, or H367IP the HP Guardian TCPIP product, is used for transport of NetEx requests and buffers between the H267IP (or H367IP) host and the TNP NetEx.

Chapter 2: NetEx/IP and the ISO Model

In creating NetEx/IP, Network Executive Software followed the guidelines set by the International Standards Organization (ISO) for Open Systems Interconnection (OSI). Open Systems Interconnection refers to the exchange of information among terminal devices, computers, people, and networks, that are open to communication with one another.

The ISO model is composed of seven layers. Each layer interacts only with adjacent layers in the model (see Table 1). By using this modular structure, the internal function of each layer is self-contained and does not affect the functioning of other layers.

Table 1. ISO Model

Layer	**Major Functions**
Application	High level description of data to be transferred and the destination involved
Presentation	Select data formats and syntax
Session	Establish session connection, report exceptions, and select routing
Transport	Manage data transfer and provide NetEx/IP-to-NetEx/IP message delivery
Network	Point-to-point transfer, error detection, and error recovery
Data Link	Data link connection, error checking, and protocols
Physical	Mechanical and electrical protocols and interfaces

Although each layer physically interacts only with adjacent layers, each layer appears to communicate directly with the corresponding layer of the other model. Figure 2 illustrates this concept.

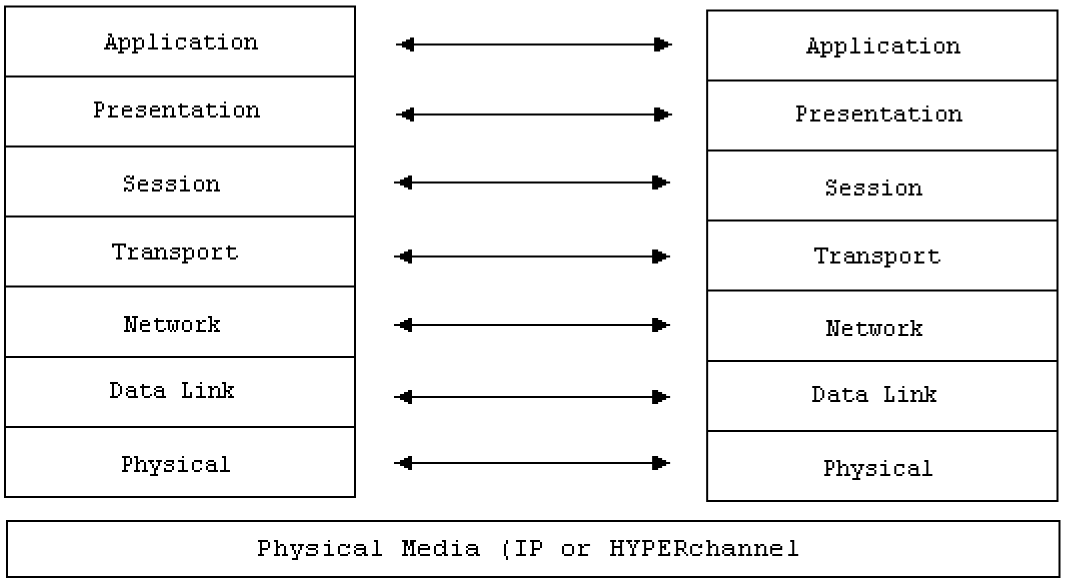


Figure 2. ISO Model Communication

Note: The corresponding layers appear to communicate directly as indicated by the lines with arrows, but actually they communicate only by progressing down through the layers of one model, through the physical media, and up through the layers of the other model.

Figure 3 shows that the hardware and firmware form the lower two layers. NetEx/IP and the driver comprise the next three layers. The NetEx/IP software provides complete session, transport, and network layer interfaces. This leaves the user free to write the application programs that use NetEx/IP or to use Network Executive Software utilities.

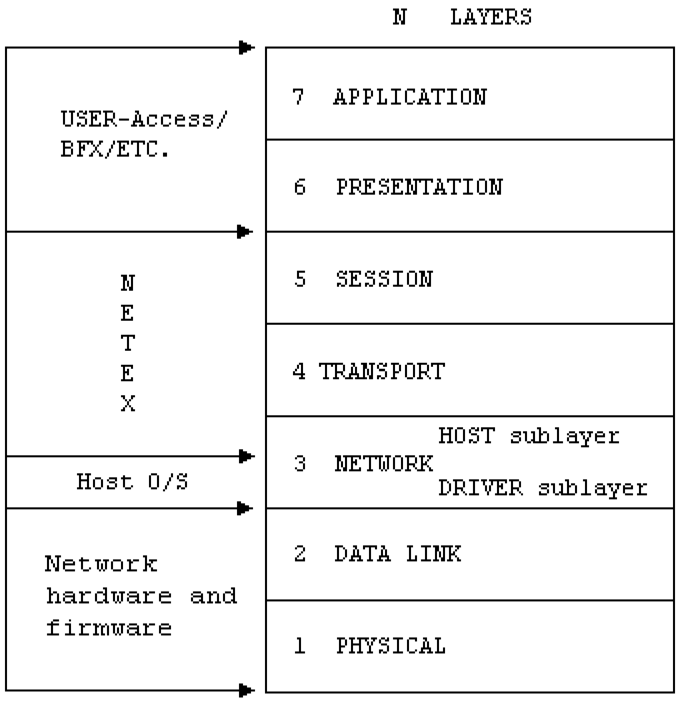


Figure 3. NetEx/IP and the ISO Model

Session Layer Services

As the highest layer within NetEx/IP (referring to the ISO model in

Figure 3), the NetEx/IP session layer software provides the general interface to the user's application/utility program. The NetEx/IP session layer services include: program-to-program connection using the best available network path, reading data, writing data, disconnection, and statistics gathering. The user requests these services using a standard NetEx/IP Request Block (NRB) (containing parameters), and the NetEx/IP requests described in "NetEx/IP Session Services". The session layer software implements user requests by requesting services from the underlying transport layer.

Transport Layer Services

The transport layer provides the actual data movement services for NetEx/IP. This is an internal layer used only by the session service code, not the end user. It transmits and receives user data, along with internal protocol information, to provide fast, efficient communications over the network. The transport layer accomplishes its function by performing services for the session layer software above it and by requesting services of the network layer below it.

The transport software manages the network path chosen by the session software. The session user does not need to be concerned with the actual hardware and software used to transmit data, nor with NetEx/IP-to-

NetEx/IP message delivery. The transport layer sets up hardware and software tables, provides buffering, and establishes linkages to manage the flow of information. Also, the protocol used by the transport layer software provides true full-duplex communications between subsystems, permitting asynchronous reads and writes. Because the transport layer provides a full-duplex operation, data can flow continuously, as long as it is being supplied by the user. This keeps the network path as full as possible and assures timely arrival of data to the user.

Network Layer Services

The network layer software provides communication path independence for the higher layers of NetEx/IP and assumes responsibility for keeping the network interfaces busy. This is an internal layer used only by the internal transport service, not the end user. The network layer formats the message to route the data through the network. If the protocol information overflows the message, the network layer splits the data transmissions into two driver requests. The network layer also multiplexes network connections over common driver connections and manages those driver connections.

Driver Sublayer Services

The driver sublayer software is the interface between the network sublayer and the physical network device. This is an internal layer used only by the internal transport service, not the end user. The driver converts network sublayer I/O for a particular network path into a form which is understandable to the devices. The driver delivers and receives network messages and associated data to and from the network adapters. The driver also allows retry and error recovery for network adapters, supports assembly/disassembly, and code conversion options, if these are provided by either endpoint and requested by the user’s data mode parameter.

Chapter 3: NetEx/IP Session Services

The user interface to NetEx/IP is a library that provides the user with access to the session and driver layer functions of NetEx/IP. Programming at the other levels of NetEx/IP (transport or network) is not supported.

To communicate using the session layer of NetEx/IP, the calling programs (that is, the programs that are calling NetEx/IP) must first establish a session connection. Once the session is established, data transfer may take place in a variety of ways, depending on the needs of the calling programs. One important feature of NetEx/IP is that the transport uses internal error checking and error recovery procedures. Once NetEx/IP accepts data, at the session or transport level, the user is assured of its delivery (with the possible exception of catastrophic failures – for example, a machine going down or a major problem with the communication line). Sessions may be terminated by either of the parties at any time, although this should be done by mutual agreement.

This chapter explains the concepts of session layer intertask communication using NetEx/IP. The following topics are discussed in this section:

- Session layer requests
- General concept of a session
- Establishing a session
- Data transfer process
- Terminating a session
- Handling multiple connections
- Satellite communication
- Error Recovery procedures
- Code Conversion

Session Layer Requests

There are eight requests used by programs to call NetEx/IP at the session level. These requests must be issued in a logical order according to rules described in the following paragraphs. These requests and a table called the NetEx Request Block (NRB) are the programmer's interface to NetEx/IP. The NRB is updated by the calling program when the program issues requests (either directly or via NetEx/IP), and it is updated by NetEx/IP when requests are completed by NetEx/IP. The NRB is described in the chapter "Chapter 4: NetEx Request Block".

The NetEx/IP session requests, listed in the approximate order in which they are issued, are as follows.

SOFFER

This command makes a calling program using NetEx/IP available to another program on either a remote or local host.

SCONNECT

This command requests a session with a calling program that previously issued an SOFFER. The program may insert values defining characteristics of the upcoming session in the NRB when this request is issued. This request may also write data to the offering program provided this data does not exceed the maximum segment size specified on either the sending or receiving NetEx/IP.

SCONFIRM

This command is the last step to establish the session. The host which initially issued the SOFFER replies to a SCONNECT with the SCONFIRM request if the characteristics of the session (for example, data block size, etcetera) specified in the SCONNECT are accepted. This request may also write data to the connecting program provided this data does not exceed the maximum segment size specified on either the sending or receiving NetEx/IP.

SWRITE

This command sends data to the other program. The SWRITE request may only be used after a session has been properly established. The SWRITE request is an asynchronous service (the user is free to continue when NetEx/IP accepts the SWRITE). A datamode may be specified to invoke code conversion and data assembly or disassembly.

SREAD

This command receives data that has been written by another program. The SREAD request is also used to accept indicators such as SCONFIRM, and DISCONNECT. The SREAD request is an asynchronous service, so the user may continue when NetEx/IP accepts the SREAD Request.

SWAIT

This command is specified as part of a request or as a separate request, SWAIT suspends processing on the issuing program until NetEx/IP completes the specified request(s). For SWRITE, SCLOSE, SCONNECT, SCONFIRM, or SDISCONNECT requests, NetEx/IP accepts the request quickly and the issuing program can consider the request complete. For SREAD and SOFFER requests, the request does not complete until the other program issues a SWRITE, SCLOSE, SCONNECT, SCONFIRM, or SDISCONNECT request.

SCLOSE

This command is the last write operation for a connection. The SCLOSE request is issued to gracefully terminate a connection. It may contain data just like a WRITE request, but it also indicates that the sender is ready to terminate the connection. After the SCLOSE is issued, incoming data may continue to be read, but no other messages may be written. When the other party in the connection issues an SCLOSE, the session is terminated.

SDISCONNECT

This command immediately terminates a connected session. The SDISCONNECT request may be issued by either program at any time.

These requests are used during the session as described in the following paragraphs.

General Concept of a Session

Before explaining in detail how each part of a session is programmed, the next paragraph explains the general concept of a simple NetEx/IP session.

Figure 4 shows a sample system configuration. Figure 5 is a simplified example of a session where one program reads data from another.

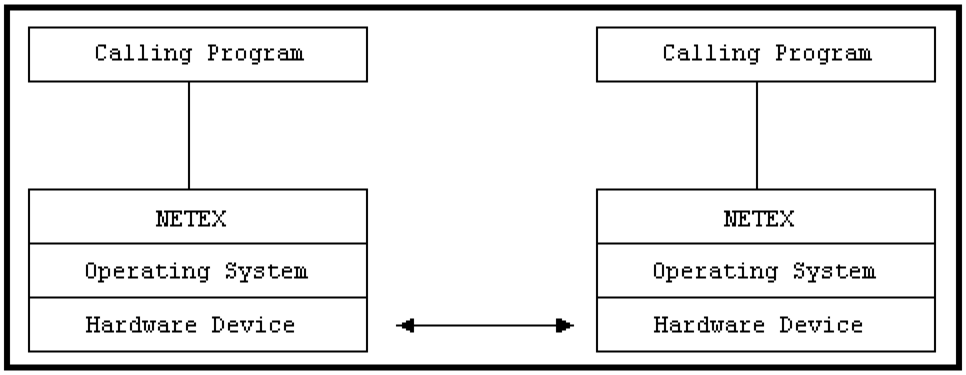


Figure 4. Sample System Configuration

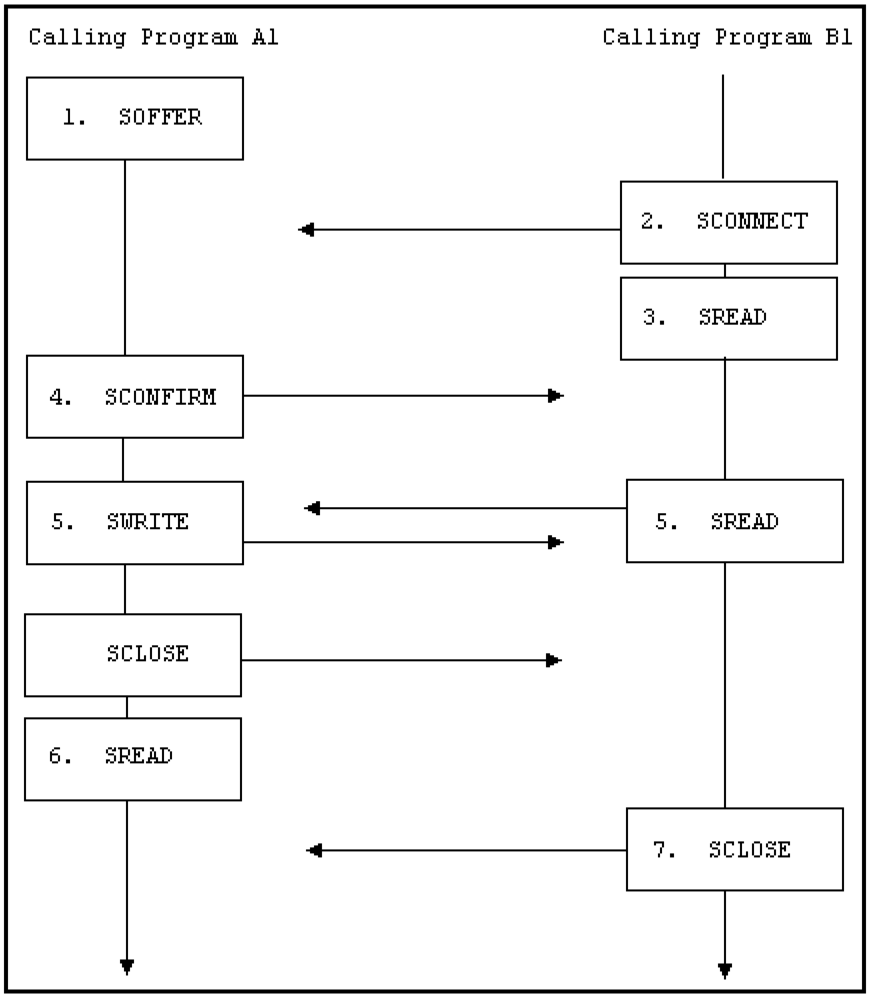


Figure 5. Simplified Session Example

The following text describes the session flow shown here in Figure 5:

1. Calling program A1 in host A issues an SOFFER indicating that it is available to service other calling programs.
2. Calling program B1 requires a file controlled by program A1. To initiate a data transfer session, program B1 issues an SCONNECT request. This request may contain data to be delivered to the offering program.

3. When the SCONNECT completes (is accepted by NetEx/IP), program B1 issues an SREAD to prepare to receive program A1's response to the SCONNECT. (Since the SCONNECT can also transport data, program B1 could issue an SDISCONNECT and terminate the session immediately. In that case, B1 would not know the status of the data transmitted.)
4. When the SCONNECT is received by the NetEx/IP in A1's host, the SOFFER completes with a Connect Indication and with B1's SCONNECT data in the buffer associated with A1's SOFFER. If program A1 finds the conditions associated with the SCONNECT acceptable, it responds by returning an SCONFIRM request.

   If the program A1 finds any conditions associated with the SCONNECT to be unacceptable, it would SDISCONNECT the session and may return data specifying the reason for the SDISCONNECT. For example, if one program determines the other program does not have the proper security code for data in that database, the session may be disconnected (SDISCONNECT).

   The SREAD that program B1 previously issued now indicated program A1's response. If program A1 issued an SCONFIRM, the SREAD will show a Confirm indication along with the data sent with the SCONFIRM. If program A1 issued an SDISCONNECT, the SREAD will complete with a Disconnect indication.
5. The programs may now begin the data transfer. Program B1 issues an SREAD to prepare to receive data from program A1. Program A1 writes data to program B1. The SWRITE command is used until the last data is written. An SCLOSE is used to write the last data. Since we are only issuing one write request in this example, the SCLOSE is used.
6. After completion of the last data transfer, program A1 issues an SREAD to detect program B1's next request.
7. Program B1 issues an SCLOSE and the session is terminated. Both programs may perform disconnect functions (for example, closing files, etcetera). Program A1 could then SOFFER itself as ready for another session.

This is a simplified example of a session. The following sections describe how to program NetEx by describing (in detail) establishing a session, data transfer and terminating a session.

Establishing a Session

Figure 6 is a flow chart showing how a connection is established using the session layer interface. Only steps which may occur in a normal process are shown in the figure. Other possibilities that are less likely to occur are discussed in the accompanying text.

Figure 6 refers to the NRB. The NRB (discussed in Chapter 4: NetEx Request Block) is a block of parameters used to signal requests to NetEx/IP and to return status to programs.

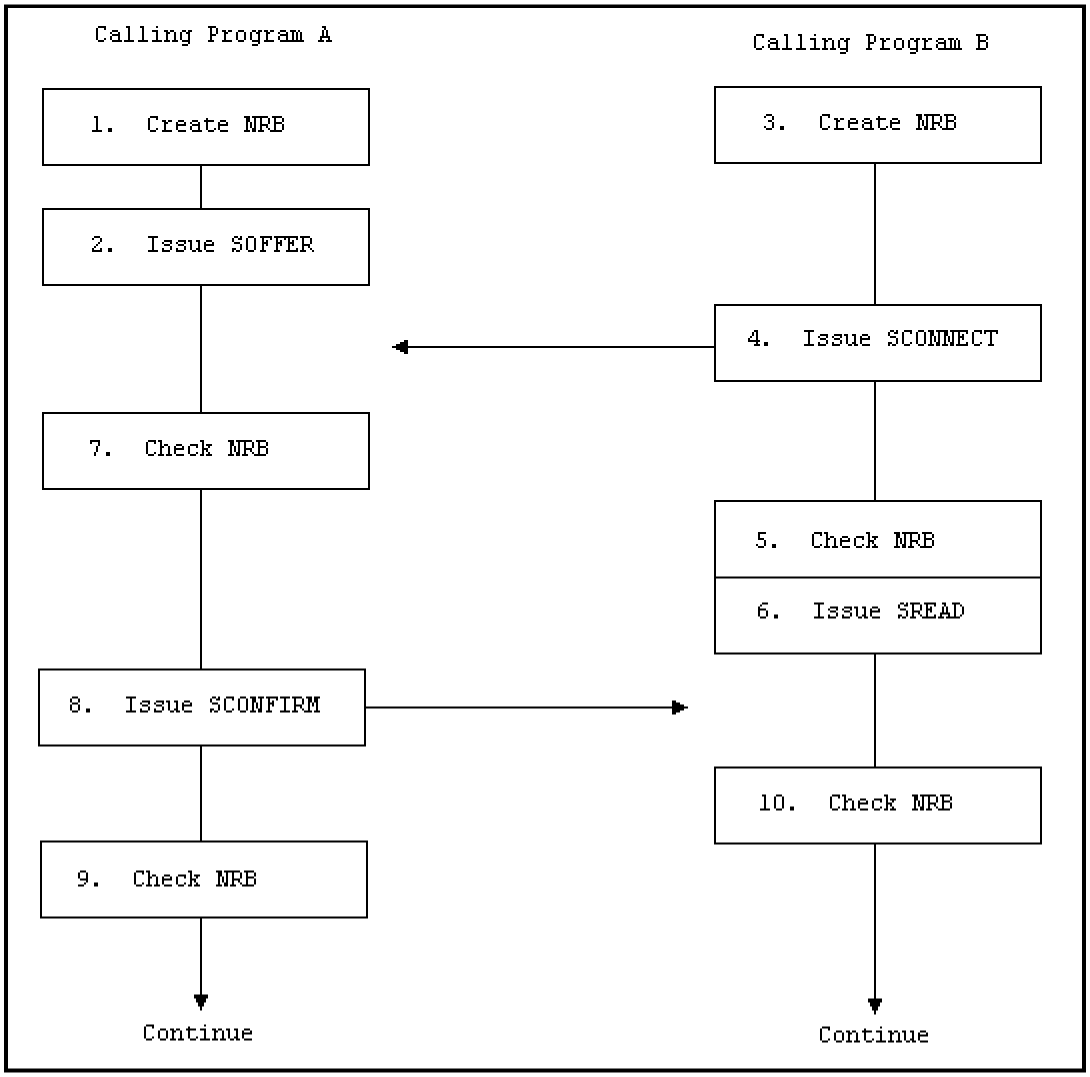


Figure 6. Establishing a Connection

The following numbered items refer to the steps in Figure 6. The steps are numbered to simplify the discussion and do not necessarily represent the exact order in which the events occur.

1. Program A prepares for the session by opening files and creating an NRB.
2. Program A issues an SOFFER to make it available to other NetEx/IP programs. The SOFFER may specify a data area for data associated with an upcoming SCONNECT.
3. Program B is a program that needs to establish a session with program A. Program B must first open files and create its own NRB.
4. Program B then issues an SCONNECT. The SCONNECT may contain such data as a password.
5. Program B then checks the NRB to determine the status of a request. The NRB indicates whether a request is in progress, whether a request has completed successfully, or whether the request has generated an error.

   Figure 6 continues assuming the NRB indicated normal completion. If the NRB indicates that a request is in progress, it would have to be rechecked after a short delay. If the NRB indicates an error, the error code would be logged and the session would not be established. The calling program should then either try again to establish a session, or should act appropriately (such as closing files that were opened before the session was attempted).

6. Program B expects program A to respond to the SCONNECT with an SCONFIRM or an SDISCONNECT. Program B issues an SREAD which would detect program A's response.
7. Program A checks its NRB to see whether the SOFFER completes. The NRB indicates that program B has issued an SCONNECT.

   If the NRB indicates that an error occurred, program A would act appropriately (which could be disconnecting from this session and reissuing the SOFFER).

   A password may be required at this time. The password could be sent as data associated with the SCONNECT. After the SCONNECT is received, the password is examined. The password could be used to restrict access to certain files, restrict access by certain programs, or have other customized uses. If a program attempts to access restricted files or has an incorrect password, program A may issue an SDISCONNECT and terminate the session.
8. If all conditions associated with the SCONNECT are acceptable, program A issues an SCONFIRM to establish the session. The SCONFIRM may contain data.
9. Program A checks the NRB to make sure that the SCONFIRM was accepted by NetEx/IP. If it was, program A would continue with the session. If NetEx/IP did not accept the SCONFIRM, program A would either retry issuing the SCONFIRM or take other appropriate action.
10. Program B checks the NRB to determine if the SREAD has successfully completed and to see what call program A issued. If program A had responded with an SCONFIRM, program B would continue with the session. If program A had responded with an SDISCONNECT, program B would have to examine the reason code associated with the SDISCONNECT, and take the appropriate action.

The previous discussion outlines the rules to be followed when establishing a NetEx/IP session. It also introduces the concept of using the NRB for communication with NetEx/IP. After requests are issued, the NRB is examined to see when NetEx/IP completes the request. This may be done using the SWAIT request, or by periodically checking the NRB fields. The NRB fields are discussed in "Chapter 4: NetEx Request Block".

Data Transfer

Unlike rules for establishing a session, NetEx/IP provides a great deal of flexibility in how session requests are used for the data transfer process. The following paragraphs describe two methods for programming data transfer. The first method is a basic data transfer technique, the second uses the more advanced technique of concurrent SWRITE and SREAD requests.

Write/Read Data Transfer

The following paragraphs describe the session requests that are used to transfer data in a straight-forward way. In general, calling programs use the SREAD and SWRITE requests to perform the data transfer. Figure 7 shows how the calling programs perform the data transfer. SWRITE requests are issued by one program which must be received by an SREAD issued by the other program.

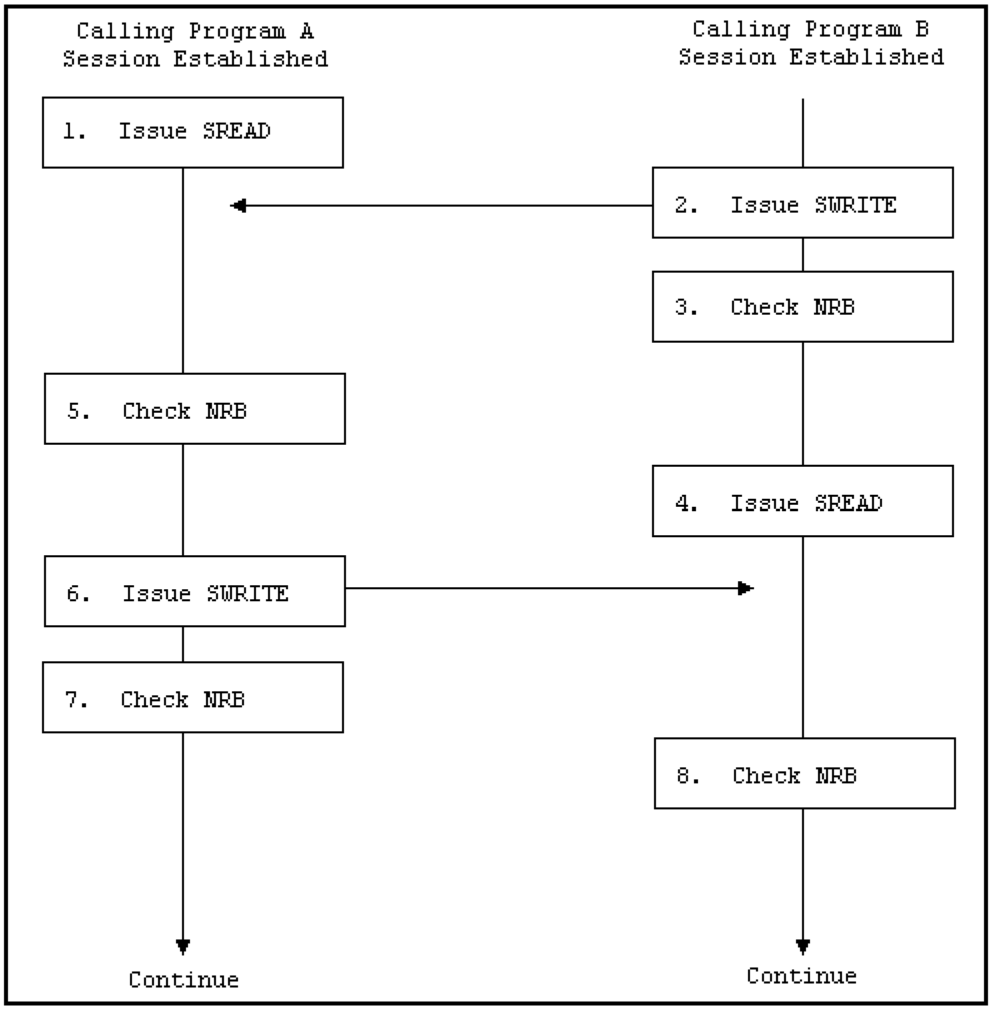


Figure 7. Write/Read Data Transfer

The following numbered items describe the steps in Figure 7. The steps are numbered to simplify the discussion and do not necessarily represent the exact order in which the events occur.

1. After a session has been established, program A issues an SREAD. The SREAD specifies the buffer for receiving data.
2. Program B issues an SWRITE. The SWRITE specifies where the data to be written is located and how long it is.
3. Program B checks the NRB to see if NetEx/IP accepted the SWRITE or indicated an error.

   Once NetEx/IP has accepted the data, it will deliver the data unless a catastrophic loss of connection occurs.
4. Program B issues an SREAD to detect program A's next request.
5. Program A received an updated NRB which indicates what program B has issued. In this case, program B has written data as program A expected. If the NRB word indicated an error, program A would act appropriately.

   If program B issued an SDISCONNECT, program A would check the reason for the SDISCONNECT and act appropriately.

6. Program A is programmed to SWRITE some data back to program B. Program A issues an SWRITE specifying the location of the data to be written.
7. Program A verifies that NetEx/IP accepted the SWRITE by checking the NRB. If the NRB indicates the SWRITE was not accepted, program A would act appropriately.
8. Program B checks the NRB and determines what program A issued.

   Both programs continue with the data transfer until they have completed their functions.

As when establishing a session, the SWAIT request may be used with other requests. Abnormal terminations are discussed later in this chapter.

Concurrent Write and Read Data Transfer

A more advanced method of data transfer uses read and write requests that are issued without expecting the other calling program to respond immediately. This type of technique is useful for long distance communications, but is also well-suited for local data transfer.

Because NetEx/IP will only accept one request using a specific NRB, two NRBs must be created by each program to perform concurrent read and writes. One NRB is used to establish the session, as previously described, and a second is created before data transfer begins. The second NRB must be created as a copy of the first to ensure that NetEx/IP internal words are preserved.

Figure 8 shows how the programs perform the data transfer. As an example of what work the program is doing, consider the following. Program A first requests data from Program B. Program B then writes data until program A writes an acknowledgement or another message. Notice that program A does not respond to every SWRITE issued by program B. When program A has received what it needs, it terminates the session using the termination procedure discussed later in this chapter.

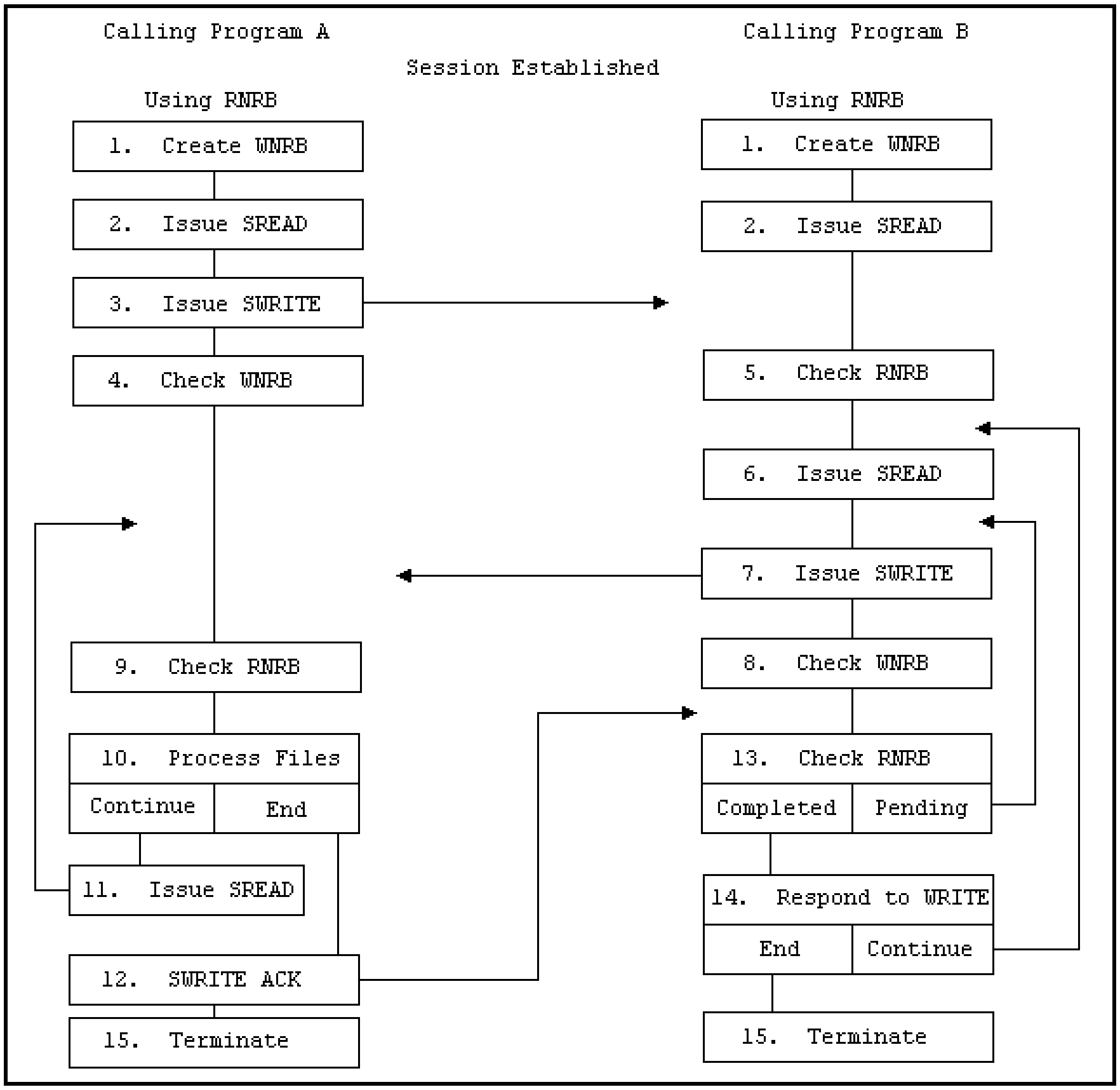


Figure 8. Concurrent SREAD and SWRITE Requests

The following numbered items describe the steps in Figure 8. The steps are numbered to simplify the discussion and do not necessarily represent the exact order which the event occur.

1. After a session has been established, both programs create duplicate NRBs for the SWRITE requests. The NRBs created before the sessions were established are assumed to have been called RNRB, and will be used with SREAD requests. New NRBs called WNRB are duplicates of the RNRB that will be used with the SWRITE requests.
2. Both programs issue an SREAD request to prepare to receive request from the other program. The RNRB is specified in the SREADs as the place for NetEx/IP to respond to that request.
3. Program A issues an SWRITE to program B. The SWRITE contains a request for specific data from program B. The WNRB is specified in the SWRITE as the place for NetEx/IP to respond to that request.
4. Program A checks the WNRB to see if NetEx/IP accepted the SWRITE or indicated an error and acts accordingly.

5. Program B, which has been checking the RNRB or waiting for it to complete, received the SWRITE from program A. This SWRITE contains parameters which program B will use to determine what data to send to program A.
6. Program B issues another SREAD that will be “floating” while processing continues.
7. Program B begins to SWRITE the data requested by program A. The WNRB is specified to monitor the SWRITE requests.
8. Program B checks the WNRB to make sure NetEx/IP accepted the SWRITE.
9. Program A checks its RNRB and discovers the SWRITE issued by program B.
10. Program A processes the files received. If program A has not yet received all the data it asked for in step 3 and wished to continue reading, it jumps to step 11. If program A wishes to respond to program B (to stop the transfer or to request other data), it jumps to step 12 and SWRITEs an appropriate message.
11. Program A issues an SREAD to continue receiving information from program B.
12. Program A SWRITEs an acknowledgement or a message to program B. Since program B has an SREAD floating, it will receive this SWRITE.
13. Program B checks the RNRB. If the SREAD has completed (meaning program A has written something), program B continues with step 14. If the SREAD is still pending (or floating), program B continues WRITING data to program A by jumping back to step 7.
14. Program B responds to program A’s SWRITE. This response could include starting to transmit other data
15. The session is terminated normally when program A has received all the data it wanted, or by special request of one of the programs.

The previous example demonstrates the technique of using SREAD and SWRITE requests when using this technique.

One-Way Data Transfer

A typical use of NetEx/IP is a one-way data transfer. Figure 9 shows how a one-way data transfer could take place. Programs A and B establish a session as described earlier in this section. Program A, which will receive data, creates a single NRB. Program B, which will send data, creates an RNRB (for monitoring SREAD requests) and a duplicate WNRB (for monitoring SWRITE requests).

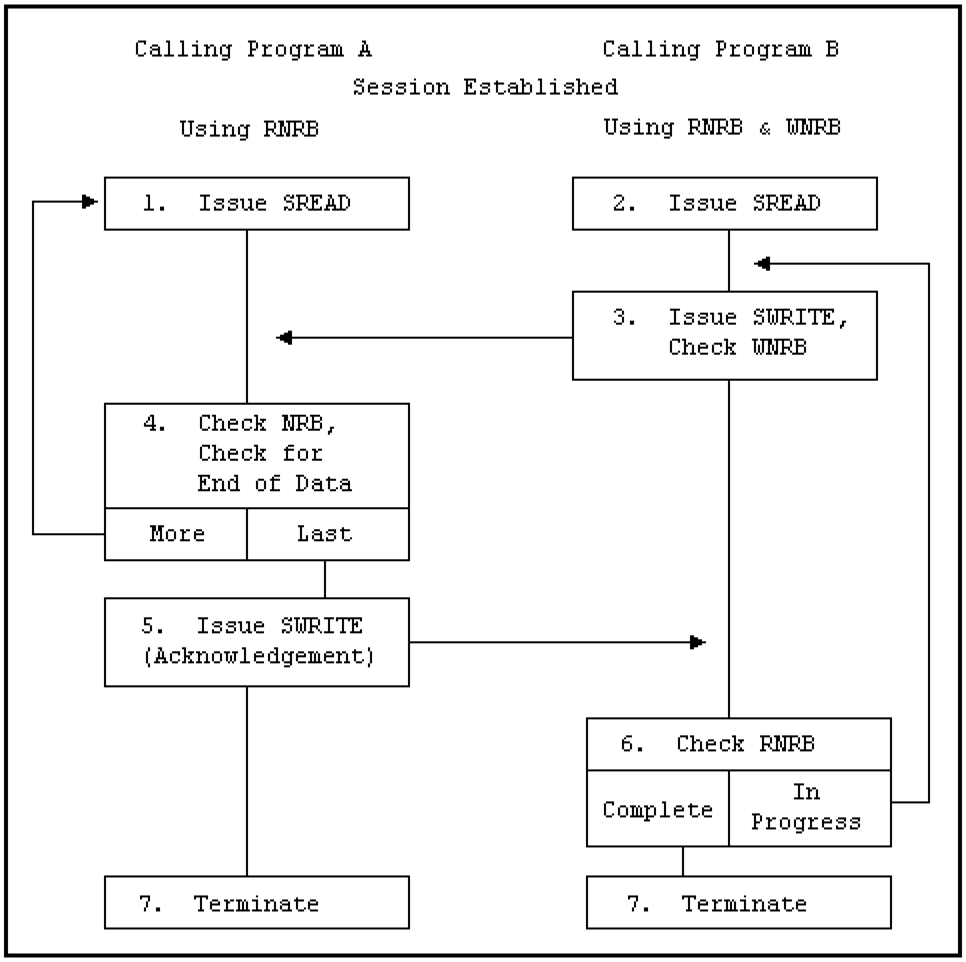


Figure 9. One-Way Data Transfer

The following numbered items describe the steps in Figure 9. The steps are numbered to simplify the discussion and do not necessarily represent the exact order in which the events occur.

1. Program A issues an SREAD request to prepare to receive data.
2. Program B issues an SREAD request to receive unexpected SWRITEs from program A. For example, if program A were unable to receive more data, it could notify program B using a previously determined set of indicators. Normally, this SREAD would not complete until all the information has been transferred.
3. Program B issues an SWRITE to transfer data to program A. An 'End of Data' indicator is placed in the data field with the last piece of information. Program B checks the WNRB to see if NetEx/IP accepted the SWRITE or indicated an error and acts accordingly.
4. Program A checks the NRB to see if the SREAD completed. If it did not complete, program A continues to check it. When the SREAD completes, program A processes the incoming information, and checks for the 'End of Data' indicator. If there is more data coming (indicator not set), program A issues another SREAD (step 1). If this is the last piece of information, program A continues with step 5.
5. Program A issues an SWRITE acknowledging that all of the information has been received. (NetEx/IP has verified the integrity of the data.)
6. Program B checks the RNRB. If it is still in progress (because program A has not written anything), program B jumps back to step 3 and SWRITEs more data. If all data has been written, program B will issue an SWAIT to suspend execution until program A returns a response. When the RNRB completes, pro-

gram B checks the data written by program A. Normally, this would be an acknowledgement of the last piece of data. In that case, program B would continue with step 7. If a problem is encountered, program B acts accordingly.

7. Program B terminates the session.

In the previous example, SWAIT requests may be used freely by program A, but should generally be used only after SWRITE requests by program B. An SWAIT could be issued by program B to wait on the RNRB after all data has been written.

Terminating A Session

NetEx/IP sessions can terminate either normally or abnormally. A normal termination is planned by the programs involved. An abnormal termination is any unplanned termination of the session.

Normal Termination

Figure 10 shows the exchange of session calls associated with normal (planned) termination. The steps are numbered to simplify the discussion and do not necessarily represent the exact order in which the events occur.

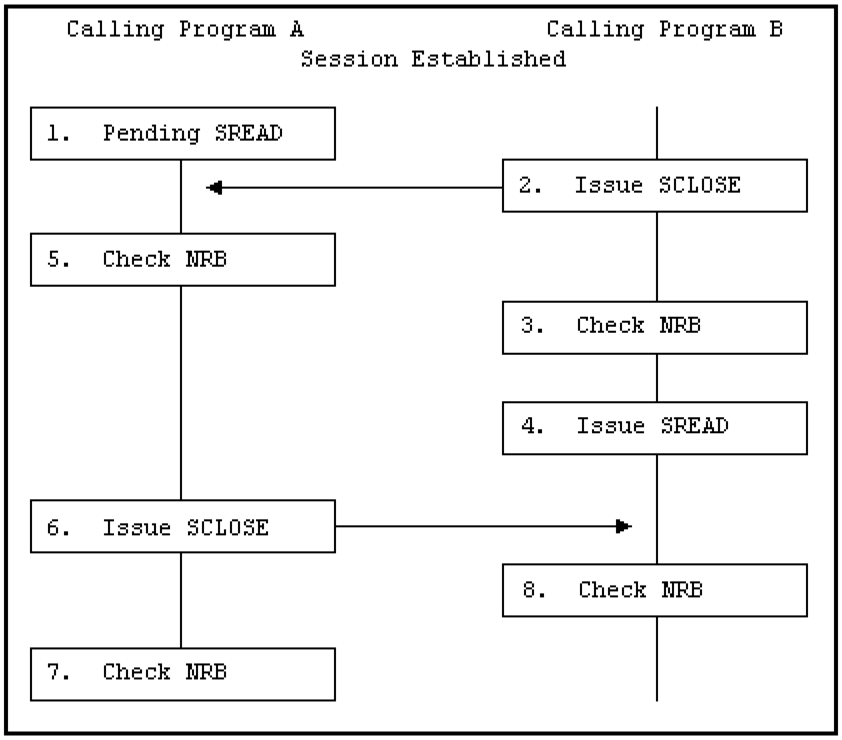


Figure 10. Normal Session Termination

The following numbered items refer to the steps in Figure 10.

1. Program A has a previously issued SREAD pending.
2. Program B issues an SCLOSE. The SCLOSE takes the same form as an SWRITE request.
3. Program B checks the NRB to verify that the SCLOSE was accepted by NetEx/IP. If it was accepted, program B may close output files or perform other termination processing. No other NetEx/IP write-type requests (except SDISCONNECT) may be issued by program B during this session. If the SCLOSE is

not accepted, Program B must act appropriately (check if NetEx/IP is down or if the other application is not there).

4. Program B issues an SREAD. Program A may still write data to program B, or program A may issue an SCLOSE or an SDISCONNECT to terminate the session.
5. Program A detects the SCLOSE.
6. Program A issues an SCLOSE to terminate the session. Data may be associated with this request. Program A may issue any number of SWRITE requests before the SCLOSE.
7. Program A checks the NRB to make sure that the SCLOSE completed successfully. Program A closes files and performs other termination processing.
8. The SREAD issued by program B completes and the NRB indicates that the session has been terminated. Program B closes files and performs other termination processing.

Abnormal Session Termination

The program must be able to react to abnormal terminations. Sessions may be abnormally terminated by the other program or by NetEx/IP.

Sessions may be unexpectedly terminated by the other program for various reasons depending on how the program is written. Some typical reasons for immediate terminations are as follows:

- A program fails to provide the proper password or authorization for a session
- A program attempts to access data that it was not authorized to access.
- The program detected an internal failure such as a program check.
- A time limit was reached
- A program encountered problems issuing a request to NetEx.

Some of these problems may be overcome by reconnecting with the calling program.

NetEx/IP may terminate a session unexpectedly because of problems with the physical network or with a host computer. This type of error may not necessarily be solved by simply reconnecting with this host. Alternatives should be provided in the calling program.

Programming Notes

The following sections provide supplemental information intended to make programming NetEx/IP simpler:

- Handling Multiple Connections
- Service Wait Options
- Satellite Communication
- Error Recovery Procedures
- Code Conversion Options

Handling Multiple Connections

NetEx/IP provides the capability for a program to be simultaneously connected to more than one other calling program. Requests coming from different connections are identified using the NRBNREF word of the NRB.

NRBNREF contains a unique number assigned to a connection when it is established. The capability to handle multiple connections enables the programmer to establish database server and requester programs.

Database server programs allow a network of hosts to use each other's databases. A database server program simply OFFERs itself to other hosts. When another host (requester) establishes a connection, the server SREADs or SWRITEs files to or from its database as specified by the requester.

Database server programs may issue multiple offers (SOFFER) by specifying a different NRB with each SOFFER. The offers (SOFFER) are completed in the order that they are issued. Many users on one machine may issue multiple OFFERs if each is generated as if it were an individual host.

Program B in the concurrent write/read example (Figure 8) is an example of a simple database server. Program B SWRITEs the files program A requests, and then waits for more instructions. More sophisticated database servers could allow themselves to connect to several requesters at one time.

Program A in Figure 8 is a simple example of a database requester.

Service WAIT Options

On each session call the user has the option to wait or not to wait for the request to complete. If the waiting is desired then the "W' form of the call should be used. The User Request Manager will issue the wait on the users' behalf and return control to the user when the NRB is posted.

The SWAIT request may be used to wait on a single NRB, a list of NRBs or may be used to wait on 0 NRBs. The way to use the wait request depends on the situation.

- If servicing a single connection where data moves logically in only one direction, use the wait option on the requests.
- If servicing multiple simultaneous connections, or a single connection where data flows both ways, use SWAIT(n) to wait on a list of NRBs.
- When servicing both NetEx/IP and real-time applications, use SWAIT(0) to wait on 0 NRBs, then check the real-time device. When issuing an SWAIT on 0 NRBs, check the NRBSTAT fields of the NRBs for the requests that you are interested in.
- NetEx/IP also needs to get control to update NRBs and send them back to the user through cross-memory services. Therefore, SWAIT(0) is important for host based NetEx.
- Another way to wait for any NRB to complete, without specifying each NRB on the call, is to use SWAIT(-1). This form of wait will return to the user only when an NRB has completed. This differs from SWAIT(0), which may return without an NRB completing.

The following points apply to waiting:

- A request cannot be marked complete until it is waited on.
- The NRB and the data buffer cannot be reused until the request is complete.

Long Latency Communications

The standard NetEx/IP requests may be used when there is long distance in the network. NetEx/IP was designed and implemented with the capability to recover from errors and lost messages using protocols which make the solving of these problems transparent to the user.

IMPORTANT: Because of the long propagation delay (, the communications channel must be kept "full" of data. This is done by using concurrent SREADs and SWRITEs which transmit data in large amounts before having the writing application stop to wait for an acknowledgement from the reading application. In this way, data is transmitted rapidly and the propagation delay has less effect on performance. (This technique requires a large buffer area.)

For example, if the calling programs acknowledge every block written in one direction with a corresponding acknowledgement written in the other direction, the propagation delay would have major impact. But, if an entire file is transmitted before an acknowledgement is returned, the effect of the propagation delay is minimized.

Minimizing the effect of the delay in this manner must be balanced with the consideration that if there is a catastrophic failure of the network, NetEx/IP, or the other host, there is no way to know how much unacknowledged data was successfully received.

Determining the frequency of the checkpoint acknowledgements is an important consideration. This decision must be made by considering the needs of individual implementations.

NetEx/IP Error Recovery Procedures

Calling programs must be able to recover from errors identified by NetEx/IP. These errors will be returned when a NetEx/IP operation does not complete successfully. The following paragraphs describe the NetEx/IP error codes and some common error recovery procedures.

Error Codes

Whenever a NetEx/IP request is issued, the results of the request are returned in one or both of two NRB fields, NRBSTAT and NRBIND. These are located at the beginning of the NRB to make their subsequent examination by high level language programs a simple matter.

NRBSTAT indicates whether an operation is in progress and whether it completed successfully or not. NRBIND indicates the type of information that arrived as the result of a read-type command (SREAD or SOFFER).

When an operation is accepted by the NetEx/IP user interface, the value of NRBSTAT is set to the local value of –1. Thus, the sign of this word is an “operation in progress” flag for all implementations.

If an operation completed successfully, NRBSTAT will be returned as all zeroes. If a read-type command was issued, then an “indication” will be set in NRBIND when the SREAD completes.

If the operation did not complete successfully, the NRBSTAT will contain a standard error code. NRBSTAT is represented as a decimal number that is potentially as large as 2^{15} –1(32,767) The 2^{16} bit is not used so that it may remain an “in progress” flag for the 16 bit machines. The error codes are listed and described in appendix A.

Common Error Recovery Procedures

The following items are some commonly encountered errors and an explanation of how to recover from them.

- Other program not there – Operators or users must coordinate the running of the two NetEx/IP programs so that one has not timed out before the other has had a chance to establish a session.
- Other program busy – Retry NetEx/IP after a suitable delay.
- NetEx/IP requests out of sequence – Sessions must be completely established before write or read requests can be issued. Sessions are established using the offer, connect, and confirm requests in that order.

Code Conversion

NetEx/IP provides for common types of code conversion by using NetEx/IP software facilities. The calling program uses the datamode (NRBDMODE) word of the NRB to specify code conversion. The caller simply specifies the source character set and the destination character set. NetEx/IP then performs code conversion using software as necessary.

Chapter 4: NetEx Request Block

The NetEx Request Block (NRB) is a block of parameters used to pass information between calling programs and NetEx/IP. The NRB is the means by which programs and NetEx/IP communicate with one another. The NRB is created by a calling program and may be updated by the program to pass information to NetEx/IP, or NetEx/IP may update the NRB to pass information to a program.

Each time a program makes a request to NetEx/IP, the program specifies an NRB to be associated with the request. NetEx/IP passes status information about that request back to the program via the NRB. Therefore, only one NetEx/IP request may use an NRB at one time. If concurrent read and write requests are used, or if a server program will be connected to more than one program at a time, several NRBs must be used.

Rules for NRB Usage

The following principles are designed to make high level language usage of NetEx/IP somewhat transportable between machines.

- Before initiating a connection, the user must clear the entire NRB structure. When the connection is initiated, the user places whichever non-default values are needed in the user part of the NRB, and invokes NetEx/IP service. Once the connection is initiated, the user must not change the OS dependent part of the NRB between calls to NetEx/IP.
- If the calling program is using the connection in a full duplex manner, the user will need to make a copy of the NRB to produce a “read NRB” and a “write NRB.” This copying operation is the copying of all 40 words of the NRB to another area, at a time when the NRB being copied is not active. If a second request for the same connection is issued from the copied NRB, the user interface must detect the condition and handle the new part of the connection accordingly.
- Many NRB values are specified in addressable units. An addressable unit is the amount of information contained in one memory location for that machine. For example, a Unisys Dorado Systems OS2200 is 36 bits, IBM systems are generally eight bits, etc.
- For multi-threaded implementations, if a session is to be shared between threads, you must insure that each thread has its own copy(ies) of the NRB made while no NetEx/IP calls for that NRB are active.

NRB Fields

Figure 11 shows the fields in the NRB. The NRB contains 40 fields. Refer to the netex.h header file for field sizes and order. All fields not defined in the table are reserved and should not be used.

Many of the NRB fields may be updated by either program or NetEx/IP with every request. However, the fields NRBCLASS, NRBMAXRT, NRBBLKI, NRBBLKO, NRBRSV, NRBOFFER, and NRBHOST are associated with the session negotiation process. Information in these fields is updated by NetEx/IP as their values change. Some of these fields are initially specified by the user in the OFFER and CONNECT requests parameters. When the CONFIRM completes, the negotiated values are returned in the NRB associated with the read of the CONFIRM information. Subsequent attempts to change these fields will have no effect.

NRB fields may be referenced by the names in the netex.h header file. Figure 11 shows the names and has a short description of these fields.

nrbstat	NetEx/IP request status returned to user

nrbind	Data type indication from OFFER/READ
nrblen	Length of data
nrbubit	Unused bit count
nrbreq	User request code
nrbnref	NetEx/IP reference number identifying connection
nrbbufa	Starting address of user's buffer
nrbbufl	Length of user's buffer
nrbdmode	Datamode for Write request
nrbtime	Request timeout in seconds
nrbclass	Class of service
nrbmaxrt	Maximum data rate permitted
nrbblki	Maximum buffer size for input requests
nrbblko	Maximum buffer size for output requests
nrbprota	Address of Odata
nrbprotl	Length of Odata
nrboffer	(Session Level) Offer name
nrbhost	(Session Level) Remote hostname
nrbuser	User ID set by some NetEx/IP APIs
nrbosd	Reserved (operating system dependent data)

Figure 11. NetEx Request Block (NRB)

The following paragraphs describe the fields in the NRB shown in Figure 11.

NRBSTAT

NRBSTAT contains a status summary of the request issued by the user. If the request is currently in progress, the entire field contains a -1 (all ones). If the request completed successfully, the NRBSTAT is 0. If the request was unsuccessful (NetEx/IP or the service routine detected an error), NRBSTAT contains an error code. The meanings of the error codes are described in Appendix A and have definitions in the ntxerror.h file.

The implementation user interface must be constructed so that a program polling NRBSTAT (to determine if the request as successful) proceeds to examine the error code if a positive value is found in NRBSTAT.

A request is marked complete only after one of the following conditions is met:

- A WAIT option was integrated into the service call.
- An SWAIT request has been issued where one of the NRBs on the SWAIT list is the NRB specified.
- Any NetEx/IP service call is issued, and the NetEx/IP service finds that the request has completed recently.

NRBIND

NRBIND indicates the type of data received in response to a read, offer, or status request. If any of those read-type requests are issued, NRBIND will always receive a nonzero value.

The values returned in NRBIND are defined as follows:

INDCONNECT (1) – Connect indication

INDCONFIRM (2) – Confirm indication

INDDATA (3) – Normal data indication

INDEXPDATA (4) – Expedited data

INDCLOSE (5) – Close indication

INDDISCONNECT (6) – Disconnect indication

INDSTATUS (7) – Status Indication

If a write-type request (write, connect, confirm, close, or disconnect) is issued, the returned value of NRBIND is usually zero. If an error is returned to the write-type request, that means the connection is broken or was never established. If an error is returned to the write-type request, then a Disconnect Indication (6) is set in NRBSTAT.

If an operation did not complete successfully, then NRBSTAT will be set to a positive, nonzero value. If NRBSTAT is nonzero, then NRBIND will have one of the following values:

- If the error results in the loss of the connection or the connection not being established in the first place, then a Disconnect Indication (6) will be in NRBIND.
- If the error means that the request could not be processed but the connection remains in effect, then NRBIND will be set to zero.

NRBLEN and NRBUBIT

NRBLEN and NRBUBIT together define the amount of useful data for input and output. NRBLEN specifies the number of addressable units that are needed to contain the data. NRBUBIT specifies the number of bits in the last addressable unit that are not significant information. This allows information to be sent on the network on a logical bit basis without damaging the data.

For example, suppose a Unisys computer wished to send exactly 75 of its 36-bit words to an IBM processor and wished it returned at a later date. The Unisys user would specify NRBLEN=75 and NRBUBIT=0. Datamode would be bit stream. NetEx/IP would record 75*36=2700 bits of information was sent over the network. The IBM user would receive the information with NRBLEN=338 (bytes) (8*338=2704 bits) and NRBUBIT=4 (bits). The IBM user could later specify the same length parameters on output and return precisely 75 words back to the Unisys system.

Note: Those programs that do not need the NRBUBIT can simply ignore its existence, knowing that simply handling the information specified by NRBLEN will ensure that all information sent by the other machine will be stored or processed.

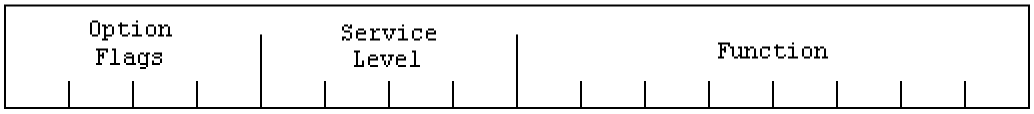
Transmitting or receiving zero-length information is possible. Zero-length data is treated as a separate transmission and is received at the other end in chronological order (as is any other data). On both the transmit and receive sides, NRBLEN will be set to zero.

If NRBUBIT is none-zero, the unused bits are not set to zero or any other value by NetEx/IP. The calling program must handle any “garbage” that may be placed in the last word of the transfer.

NRBREQ

NRBREQ is generally filled in by the API when the NetEx call is made and contains the type of request (example: SREAD) that NetEx/IP is to perform.

NRBREQ has the following format:



Option Flags refers to optional processing that NetEx/IP will perform on the request. These flags are bit significant. The bits are assigned (represented as hexadecimal numbers) as follows:

0xxx - Normal processing. NetEx/IP will return control to the caller as soon as it has internally queued the request.

1xxx to 7xxx - Reserved

8xxx - WAIT. NetEx/IP or the NetEx/IP user interface is not to return control to the user program until the request is complete.

9xxx to Fxxx - Reserved

Service Level indicates whether the request is a SESSION, TRANSPORT, NETWORK, or DRIVER type of request. Currently only the following values are supported (in hexadecimal):

x0xx - Session request

Function indicates the specific type of request to be issued. The values are assigned (in hexadecimal) as follows:

xx01 - Connect

xx02 - Confirm

xx03 - Write

xx04 - Reserved

xx05 - Close

xx06 - Disconnect

xx07 to xx80 - Reserved

xx81 - Offer

xx82 - Read

xx83 - Status

xx84 to xxFF - Reserved

The total request code is produced by combining the Option, Function, and Service Level. For example, consider SREAD with wait processing. Wait processing is 8xxx, SREAD is a x0xx Service Level plus a xx82 Function. This totals a 8082_{16} request code.

NRBNREF

NRBNREF is an internal NetEx/IP identifier that distinguishes this connection from all others maintained by this copy of NetEx/IP. This value is assigned by NetEx/IP when a connection is established.

NRBBUFA

NRBBUFA contains the start of the data buffer to be used for either input or output requests. The user must supply a valid buffer address before each input or output request. For a write request, the contents of this buffer must be left unchanged until the associated NetEx/IP write type request has completed. If a read request is issued, then the contents of the buffer should be examined when the read request completes successfully.

NRBBUFL

On input, NRBBUFL specifies the maximum size of the Pdata (ordinary data) that NetEx/IP can store in the buffer. This field is effectively ignored on output (NRBLEN and NRBUBIT are used to determine the actual length of output data). This usage difference allows a NetEx/IP user to associate an NRB with a single buffer and never change this field even if many READs and WRITEs are issued. NRBBUFL is specified in addressable units.

NRBDMODE

NRBDMODE is specified by the transmitting NetEx/IP application on any write-type request (connect, confirm, write, close, or disconnect). It is always specified as a 16-bit quantity. Datamode is forwarded through all layers of NetEx/IP. When the receiving entity received the data, the datamode specified by the transmitter (with possible modifications as described below) is inserted into the NRB associated with the receiving SREAD or SOFFER request.

Currently NetEx/IP supports auto datamode.

Auto Datamode

Auto datamode is designed for all common NetEx/IP transfers. When auto datamode is selected, the user identifies the source and destination character sets, and NetEx/IP selects the appropriate assembly/disassembly and code conversion. NetEx/IP will perform code conversion only when the selected conversion is meaningful to the receiving machine

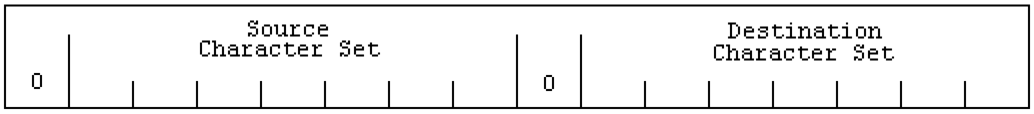
Auto datamode supports three conversion options.

Bit Stream where the bit pattern sent is precisely reproduced in the destination machine.

Octet where eight-bit binary quantities are sent from one machine to another, using an eight-bit byte representation appropriate to each machine.

Character where character information is sent from one machine to another with a full range of character assembly and code conversion options.

The conversion options are selected in the NRBDMODE field. The auto datamode has the following format in the NRBDMODE field:



'0' in the high order bit is the auto mode indicator.

Source Character Set indicates the conversion option of the data used in the write buffer of the transmitter.

'0' in the high bit of the low order byte is reserved.

Destination Character Set indicates the conversion option of the data going to the destination program. For example, a conversion from EBCDIC (3) to ASCII (2) would be entered as a hexadecimal value of 0302.

Value	Conversion Option
0	Bit stream mode
1	Octet Mode
2	ASCII (8 bit)
3	EBCDIC
4	Reserved
5	BCD
6	Field-data (Unisys)

- If the incoming driver determines that the destination character set is not "meaningful" on it own host, then it treats the incoming character set as "octet mode" data and provides the user with the data along with an error code in NRBSTAT indicating that the data was "damaged."

NRBTIME

NRBTIME specifies the length of elapsed time that the associated read-type command is to remain in effect. If a time interval equal to the number of seconds in NRBTIME has elapsed, and no data or connection information has arrived to satisfy the READ or OFFER, then the request will end with an error.

If the value in NRBTIME is "0", then the request will wait indefinitely.

NRBCLASS

NRBCLASS is a connection-negotiation parameter that defines the class of service (the type of protocol that will be used by the connection service). The current definition of class is as follows:

0	Use class determined by the Network Configuration Table (NCT).
2	Use Version 2 NetEx/IP protocol. This protocol is used in Release 2 and above NetEx/IP products.
4	Use Version 4 NetEx/IP protocol. This protocol is introduced in NetEx/IP Versions 6 and above.

All other values (or values that are not supported for a particular implementation) will return a "class not implemented" error.

The user may set this prior to the OFFER or CONNECT. When the offer or connect completes, the value of this field should contain the protocol version actually negotiated.

NRBMAXRT

NRBMAXRT (maximum rate) is a connection negotiation parameter that indicates the maximum data rate possible for the connection. If this is set in the NRB by the user, the NRBMAXRT value is the lesser of this value or maxkbitspersec value (specified in the NTXDEFAULT). This field is designed for those applications that wish to make limited use of communications media between destinations.

The units of this field are expressed as a 16-bit positive quantity giving the speed of the network in 1000's of bits per second.

Note: The NRBMAXRT and the throttling concept only directly apply to the transmitting portion of a given connection. The other party in the connection may be working with completely different throttling parameters and the corresponding program will have no direct way of knowing the remote transmitter's data rate parameters.

On completion of the offer or confirm request, this field will have a nonzero value that contains the maximum throughput that is possible to the connection, based on the user's original request and the characteristics of the communications path between the two.

NRBBLKI and NRBBLKO

NRBBLKI and NRBBLKO are connection negotiation parameters that specify the maximum amount of data that the calling program expects to read or write at one time during the coming connection. This parameter should be provided with the connect or offer request. During the session negotiation process, the NRBBLKI of one program will be compared with the NRBBLKO (output maximum buffer size) specified at the other end, and the lesser of the two values will be returned in the two respective fields.

For the connecting program, the negotiated results will be returned in the NRB along with the confirm data read following the connect. The offering program will receive the negotiated values on completion of the offer and hence may decide if the negotiated values are acceptable for the work at hand.

The NetEx/IP installation systems programmer must supply two values controlling these buffer sizes in the NTXDEFAULT file. First, the default input and output block sizes to be used if these are not specified (left zero) by the caller. Secondly, the maximum input and output block sizes permitted by the installation.

As an example of the block negotiation process: Program A issues a connect with NRBBLKI=256 and NRBBLKO=4096. The offering program B to which A will connect specifies 64K for both, allowing the connector to set any reasonable value in these fields. When the offer completes, B sees NRBBLKO=256 and NRBBLKI=4096, the minimum of the two sets of values. When A's read following the connect completes, it will see NRBBLKI=256 and NRBBLKO=4096, which are the same values as B with the directions reversed.

Two default options are available with these fields. If a zero is specified (in connect or offer) in either one of these fields, then the value used for negotiation will be an installation supplied default that is provided at NetEx/IP installation time (NTXDEFAULT). If the value in this field is the machine representation of –1, then the size used for negotiation will be the maximum size available for that installation, which is also a parameter specified at initialization time (NTXDEFAULT). Note that the values implied using zero or –1 will be used for negotiation of the connection block sizes. The actual size negotiated will be supplied in these fields on completion of the connect or offer.

NRBPROTA and NRBPROTL

The NRBPROTA and NRBPROTL are parameters that permit the application to provide Odata to the remote application. NRBPROTA specifies the address of the buffer containing the Odata, and NRBPROTL specifies the number of octets of Odata in that buffer.

When a write-type command is issued, the Odata provided (if any) will be added to the message, and eventually delivered as Odata to the receiving application's read type command. As a result, this is a second buffer that is handled in a similar way to the Pdata that is specified by NRBBUFA and NRBLEN/NRBUBIT. There are some distinct differences that are as follows:

Odata is always sent and received in "octet mode," which means it will be represented in the best way that the particular host can handle strings of eight-bit binary quantities, (for example, 1/byte, 4/36-bit word, etc.).

The maximum amount of Odata that may be sent is limited. The maximum is defined in the netex.h header. Each version of NetEx/IP will have a generated maximum on the number of bytes of Odata that it is prepared to accept in incoming messages. The maximum amount of Odata that may be sent or received will be the minimum of the Odata sizes generated on each host.

Users should be warned that sending excessive amounts of Odata with normal transmissions may result in a multiple network messages, which will increase network traffic and decrease network performance, often by a factor of two.

On a write-type operation, no Odata will be sent if NRBPROTL is zero. If a non-zero length is specified, then the Odata will be transmitted along with the Pdata, if present. When the read takes place, the Odata will be placed in the address specified by NRBPROTA and its incoming length will be set in NRBPROTL.

Always, NRBPROTL contains the length of the Odata in octets, not “addressable units”.

NRBOFFER

NRBOFFER is a required parameter for connect and offer, which specifies the offered name used to match when the offer and connect requests meet). Names of all processes are compared as uppercase alphanumeric data that are up to eight characters in length. Names less than eight characters long will be padded with blanks. Process names will be converted to the ASCII character set for transmission between hosts, so only those characters that are meaningful in ASCII should be used during the name matching process.

NRBHOST

NRBHOST is a required parameter for connects which specifies the NetEx hostname or groupname of the host computer that will be addressed to match an offer request. Names of all hosts are specified by the NCT (input for the Configuration Manager). All host names are uppercase alphanumeric data that are up to eight characters in length. Names less than eight characters long will be padded with blanks.

NRBUSER

This field may be used differently on each platform, however for H300e it is used to pass the user id through the API for display information purposes. The contents of these fields are maintained by NetEx/IP during a session.

NRBOSD

NRBOSD is reserved for internal use. NetEx/IP software uses this field to service and monitor the progress of NRB requests. The contents of these fields are maintained by NetEx/IP during a session.

If the NRBOSD field is altered by the calling program, the results are unpredictable.

Creating an NRB

A single NRB should be created before a calling program OFFERs or CONNECTs to another program. The NRB is 40 fields long and should initially be zero-filled. Programs may create several NRBs initially.

If several NRBs are required to service a single connection, they should be duplicated from the initial NRB, as described in the following paragraphs.

Duplicating an NRB

Duplicating NRBs is necessary when using multiple NRBs within a session. By duplicating the NRB, the connection-negotiation parameters, the connection reference number and the internal NRBOSD information is preserved, allowing the duplicate NRB to be valid.

To duplicate an NRB, wait until the initial CONNECT or OFFER has completed successfully, then copy the entire “working” NRB (up to and including the NRBOSD field) to a blank NRB at a different location. The second NRB can now be used for NetEx/IP requests. The NRB should only be duplicated when it is not in use by NetEx/IP; that is, when NRBSTAT is 0.

Chapter 5: C High Level Interface

NetEx includes a library of subroutines that are designed to be called by the C high level object module. Also included is a library of copy files that may be included (#INCLUDE) into a C program and inserted at compile time. When the user makes a call to the user interface, the appropriate information is supplied in parameter format to pass to NetEx.

There are two components that are used to establish working communications through NetEx: one or more NetEx Request Blocks (NRBs) that must be supplied by the C caller, and the NetEx-provided subroutines that are used to invoke NetEx services. The NRB is described first, followed by the calls to the subroutines. The calls are presented in the approximate order in which they are used:

`soffr`	`offer services`
`sconn`	`connect to an offered program`
`sconf`	`confirm acceptance of connect`
`sread`	`read incoming request or data`
`swrit`	`write data`
`sclos`	`write last data`
`swait`	`wait for previous request(s) to complete`
`sdisc`	`immediate disconnect`

C NetEx Request Blocks

The C user builds an NRB by declaring it to be a structure of type nrb. Various fields of this structure will hold the information to be transferred to NetEx, and others will contain the information that is returned by NetEx. If more than one NRB will be required to service an application, one NRB type should be defined and several NRB variables should be declared to be of that type. Before these NRB structures are used for any NetEx request, Network Executive Software advises to zero all the members of the record. This will allow defaults for fields not explicitly used by the caller to take effect.

The NetEx C functions have the philosophy that arguments commonly passed to NetEx (such as a data buffer address) will be passed as parameters to the function. “Exotic” elements to be passed to NetEx, such as maximum input block size; will be supplied by storing the desired value in the proper member of the NRB structure. When the request completes, the user C program directly accesses the desired members of the NRB structure to determine if the operation completed properly.

SOFFR C Function

The SOFFR (offer) function provides a means for a program desiring to accept a connection from a caller on the network to post the availability of the service. The SOFFR is actually a specialized form of read request. The SOFFR reads any data associated with the SCONN.

Before issuing an SOFFR function, the user must provide an NRB (described in Chapter 4: NetEx Request Block) to be used by the user interface. Also, NetEx must be active in the system.

SOFFR Function Format

The SOFFR function has the following format:

Function (Select One)	**Required Parameters**
`soffr` `soffrw`	`(nrb, buffer, length, timeout, pname)`

SOFFR Parameters

The following parameters are used in the SOFFR function. The parameters must be specified in the order presented. All parameters are required.

soffr

soffrw

This is the verb for this function. SOFFRW specifies that the calling program must wait for the call to complete before processing is resumed.

nrb

This required parameter is a pointer to the NRB data area to be passed to NetEx.

buffer

This required parameter is a pointer to the buffer data area to receive data sent by the corresponding SCONNECT request.

length

This required parameter is the length of the buffer (in addressable units) that will hold the data sent by the corresponding SCONNECT. When called, *length* should contain the maximum size of the buffer. On return, the NRBLEN (NRB.length) field will contain the number of bytes of information actually sent to the offering application. The data type of length should be INT.

timeout

This required parameter is the number of seconds that the OFFER request should remain outstanding. If no application connects during this interval, then the OFFER will end abnormally. If *timeout* is specified as zero, the OFFER will remain outstanding indefinitely.

pname

This required parameter is the alphabetic name of the process to be offered to the corresponding calling program. The name offered is arbitrary, but must be known to the connecting program. This name may be provided as a packed array(.1..8.) of character or as a string in the CALL statement, if padded with blanks to eight characters in length.

SOFFER Entry Parameters

The following NRB fields are used by SOFFR on entry.

NRBBUFA	Address for incoming Pdata
NRBBUFL	Length of buffer to hold Pdata
NRBPROTA	Address for incoming Odata
NRBPROTL	Length of buffer to hold Odata
NRBTIME	Number of seconds offer outstanding
NRBBLKO	Maximum transmission size acceptable
NRBBLKI	Maximum reception size acceptable
NRBMXRAT	Limit on transmission speed
NRBOFFER	Application name to offer

SOFFR Results

The following NRB fields are updated when SOFFR completes.

NRBSTAT	Success/failure code
NRBIND	Contains Connect Indication
NRBLEN	Length of incoming Pdata
NRBUBIT	Unused bit count of Pdata
NRBPROTL	Length of Odata received
NRBNREF	S-ref assigned this connection
NRBBLKO	Maximum transmission Pdata Size
NRBBLKI	Maximum reception Pdata size
NRBMXRAT	Maximum transmission speed of path
NRBHOST	Name of host where SCONN originated

SCONN C Function

The SCONN (connect) function provides a means for a program to request a session with a program that has issued an SOFFR. The SCONN is a specialized form of a write request. The SCONN initiates the session and may optionally write data to the offerer at the same time, provided this data does not exceed the maximum segment size specified on either the sending or receiving NetEx.

Before issuing the SCONN, an NRB must be provided for use by the user interface.

SCONN Function Format

The SCONN function has the following format:

Function (Select One)	**Required Parameters**
`sconn` `sconnw`	`(nrb, buffer, length, datamd, pname, hname)`

SCONN Parameters

The following parameters are used in the SCONN function. The parameters must be specified in the order presented. All parameters are required.

sconn
sconnw

This is the verb for this function. SCONNW specifies that the calling program must wait for the call to complete before processing is resumed.

nrb

This required parameter is a pointer to the NRB data area to be passed to NetEx.

buffer

This required parameter is a pointer to the buffer data area that holds the user data to be sent to the corresponding application

length

This required parameter is the length of the data (in addressable units) to be sent to the corresponding application, to be presented with the completion of the corresponding application's OFFER request. If no data needs to be sent to the other application, the length may be set to zero. The data type of *length* should be INT.

datamd

This required parameter is the datamode to be used to send the connect data to the corresponding application. The data type of *datamd* should be INTEGER. Refer to NRBDMODE in Chapter 4: NetEx Request Block for a discussion of the datamode parameter.

pname

This required parameter is the alphabetic name of the process offered (SOFFR) by the corresponding calling program. The name offered is determined by the other calling program. This name may be provided as a packed array(.1..8.) of character or as a string in the call invocation, if padded with blanks to eight characters in length.

hname

This required parameter is the alphabetic name of the host computer to be accessed to determine if the correct SOFFR is available. The “names” of the host computers in the network are determined by the NetEx installation systems programmer. This may be provided as a packed array(.1..8.) of character or provided as a string in the call invocation, if padded with blanks to eight characters in length.

SCONN Entry Parameters

The following NRB fields are used by SCONN on entry.

NRBBUFA	Address of outgoing Pdata
NRBLEN	Length of outgoing Pdata
NRBUBIT	Pdata unused bit count
NRBDMODE	datamode of Pdata
NRBPROTA	Address of outgoing Odata
NRBPROTL	Length of outgoing Odata
NRBBLKO	Maximum transmission size acceptable
NRBBLKI	Maximum reception size acceptable
NRBMXRAT	Limit on transmission speed
NRBHOST	Alphanumeric “host” name
NRBOFFER	Alphanumeric “application” name

SCONN Results

The following NRB fields are updated when SCONN completes.

NRBSTAT	Success/failure code
NRBIND	Set to zero
NRBNREF	S-ref (Session ID) assigned
NRBBLKO	Maximum transmission Pdata size
NRBBLKI	Maximum reception Pdata size
NRBMXRAT	Maximum transmission speed of path

SCONF C Function

The SCONF (confirm) function provides a means for an offering program to confirm (to the connector) that the connection has been successfully completed. A negative response to an SCONN would be an SDISC.

The SCONF is a specialized form of write request. Data may optionally be written during the confirm process with this command, provided this data does not exceed the maximum segment size specified on either the sending or receiving NetEx.

Before issuing the SCONF call, an SOFFR must have completed successfully by receiving an SCONN response. The calling program must provide a NRB with an NRBNREF relating to this session.

SCONF Function Format

The SCONF function has the following format:

Function (Select One)	**Required Parameters**
`sconf` `sconfw`	`(nrb, buffer, length, datamd)`

SCONF Parameters

The following parameters are used in the SCONF function. The parameters must be specified in the order presented. All parameters are required.

sconf
sconfw

This is the verb for this function. SCONFW specifies that the calling program must wait for the call to complete before processing is resumed.

nrb

This required parameter is a pointer to the NRB data area to be passed to NetEx.

buffer

This required parameter is a pointer to the buffer data area that holds the user data to be sent to the corresponding application.

length

This required parameter is the length of the data (in addressable units) to be sent to the corresponding application, to be presented with the completion of the corresponding application's SREAD request. If no data needs to be sent to the other application, the length may be set to zero. The data type of *length* should be INT.

datamd

This required parameter is the datamode to be used to send the connect data to the corresponding application. The data type of *datamd* should be INTEGER. Refer to NRBDMODE in Chapter 4: NetEx Request Block for a discussion of the datamode parameter.

SCONF Entry Parameters

The following NRB fields are used by SCONF on entry.

NRBBUFA	Address of outgoing Pdata (move mode)
NRBLEN	Length of outgoing Pdata
NRBUBIT	Pdata unused bit count
NRBDMODE	datamode of Pdata
NRBPROTA	Address of outgoing Odata
NRBPROTL	Length of outgoing Odata

SCONF Results

The following NRB fields are updated when SCONF completes

NRBSTAT	Success/failure code
NRBIND	Set to zero

SREAD C Function

The SREAD function provides a means for a program to receive data from another host or an indication from NetEx of an abnormal condition with the connection.

Before an SREAD can be issued, a connection must be established. The NRB specified must have been used for a previous NetEx request for the particular connection, or a copy of another NRB that has been used to service the desired connection.

IMPORTANT: Keep an SREAD request outstanding to receive incoming data. NetEx will automatically terminate a connection if a request is waiting to be read for too long. This read-timeout value is set at installation time.

Defaults for unspecified parameters are assumed to be the parameters existing in the NRB. Thus, after *buffer* and *length* are agreed on during the connection process, these parameters can be omitted.

SREAD Function Format

The SREAD function has the following format:

Function (Select One)	**Required Parameters**
`sread` `sreadw`	`(nrb, buffer, length, timeout)`

SREAD Parameters

The following parameters are used in the SREAD function. The parameters must be specified in the order presented. All parameters are required.

sread
sreadw

This is the verb for this function. SREADW specifies that the calling program must wait for the call to complete before processing is resumed.

nrb

This required parameter is a pointer to the NRB data area to be passed to NetEx.

buffer

This required parameter is a pointer to the buffer data area to receive the data sent by the corresponding application's SWRITE or SCONF request.

length

This required parameter is the length of the buffer (in addressable units) to hold the data sent by the corresponding SWRITE. When called, *length* should contain the maximum size of the buffer. On return, the actual length will be in NRBLEN. Programs that wish to work with the Unused Bit Count on input should examine the NRBUBIT field. The data type of *length* should be INT.

timeout

This required parameter is the number of seconds that the READ request should remain outstanding. If the corresponding application does not send data during this interval, then the read will end abnormally. If *timeout* is specified as zero, the READ will remain outstanding indefinitely.

SREAD Entry Parameters

The following NRB Fields are used by SREAD on entry.

NRBBUFA	Address for incoming Pdata (move mode)
NRBBUFL	Length of buffer to hold Pdata
NRBPROTA	Address for incoming Odata
NRBPROTL	Length of buffer to hold Odata
NRBTIME	Number of seconds offer outstanding

SREAD Results

The following NRB fields are updated when SREAD completes.

NRBSTAT	Success/failure code
NRBIND	Contains Connect Indication
NRBLEN	Length of incoming Pdata
NRBUBIT	Unused bit count of Pdata
NRBPROTL	Length of Odata received
NRBBLKO	Maximum transmission Pdata size (On Read of Confirm only)
NRBBLKI	Maximum reception Pdata size (On Read of Confirm only)

SWRIT C Function

The SWRIT (write) function provides a means for a program to transmit data to another calling program.

Before an SWRIT can be issued, a connection must be established.

SWRIT Function Format

The SWRIT function has the following format:

Function (Select One)	**Required Parameters**
`swrit` `swritw`	`(nrb, buffer, length, datamd)`

SWRIT Parameters

The following parameters are used in the SWRIT function. The parameters must be specified in the order presented. All parameters are required.

swrite
swritw

This is the verb for this function. SWRITW specifies that the calling program must wait for the call to complete before processing is resumed.

nrb

This required parameter is a pointer to the NRB data area to be passed to NetEx

buffer

This required parameter is a pointer to the buffer data area that holds the user data to be sent to the corresponding application.

length

This required parameters is the length of the data (in addressable units) to be sent to the corresponding application. The length may be set to zero. The data type of *length* should be INT.

datamd

This required parameter is the datamode to be used to send the connect data to the corresponding application. The data type of *datamd* should be INT. Refer to NRBDMODE in Chapter 4: NetEx Request Block for a discussion of the datamode parameter.

SWRIT Entry Parameters

The following NRB fields are used by SWRIT on entry.

NRBBUFA	Address of outgoing Pdata
NRBLEN	Length of outgoing Pdata
NRBUBIT	Pdata unused bit count
NRBDMODE	Datamode of Pdata
NRBPROTA	Address of outgoing Odata

NRBPROTL	Length of outgoing Odata

SWRIT Results

The following NRB fields are updated when SWRIT completes.

NRBSTAT	Success/failure code
NRBIND	Set to zero

SCLOS C Function

The SCLOS (close) function provides a way for a program to transmit data to another corresponding calling program and indicate that this is the last data to be sent. The data must be received by a read request in the other program.

After a program has issued an SCLOS, no other data may be written by that program. If the other program had previously issued an SCLOS, the data is written and then the connection is disconnected. If the other program has not issued an SCLOS, it is still free to write data to the program that did issue the SCLOS.

Before issuing the SCLOS, a connection must be fully established.

SCLOS Function Format

The SCLOS function has the following format:

Function (Select One)	**Required Parameters**
`sclos` `sclosw`	`(nrb, buffer, length, datamd)`

SCLOS Parameters

The following parameters are used in the SCLOS function. The parameters must be specified in the order presented. All parameters are required.

sclos
sclosw

This is the verb for this function. SCLOSW specifies that the calling program must wait for the call to complete before processing is resumed.

nrb

This required parameter is a pointer to the NRB data area to be passed to NetEx.

buffer

This required parameter is a pointer to the buffer data area that holds the user data to be sent to the corresponding application.

length

This required parameter is the length of the data (in addressable units) to be sent to the corresponding application. The length may be set to zero. The data type of *length* should be INT.

datamd

This required parameter is the datamode to be used to send the connect data to the corresponding application. The data type of *datamd* should be INT. Refer to NRBDMODE in Chapter 4: NetEx Request Block for a discussion of the datamode parameter.

SCLOS Entry Parameters

The following NRB fields are used by SCLOS on entry.

NRBBUFA	Address of outgoing Pdata

NRBLEN	Length of outgoing Pdata
NRBUBIT	Pdata unused bit count
NRBDMODE	Datamode of Pdata
NRBPROTA	Address of outgoing Odata
NRBPROTL	Length of outgoing Odata

SCLOS Results

The following NRB fields are updated when SCLOS completes

NRBSTAT	Success/failure code
NRBIND	Set to zero
NRBBUFA	Contains zero (if ptr mode selected)

SWAIT C Function

The SWAIT function provides the means to wait for the completion of requests that have not been previously waited for. Control will be returned to the SWAIT caller when any one of the NRBs specified no longer has the “in progress” flag set. Return from the subroutine will be immediate if any one of the NRBs has completed before the SWAIT call. By waiting on 0 NRBs, the NetEx subroutine library will take control and update all NRBs.

SWAIT(0) will return after checking all of the NRBs that are pending. It may return before any NRB has completed. Therefore, Network Executive Software advises to check the NRBSTAT after an SWAIT(0) and send another SWAIT(0) if the status is –1. This behavior is true whether SWAIT(0) is used in a single or multi-threaded application.

When SWAIT(-1) is called, NetEx takes control and checks all NRBs, just as SWAIT(0) does. SWAIT(-1) will only return to the user when at least one NRB has completed.

Note: In a multi-threaded application, the NRB which completed may not belong to the thread making the swait(-1) call. If your thread has no NRBs active, you may end up waiting for another thread’s NRB to complete! Think globally for this form of the call and use with caution!

In a multi-threaded environment, `swait(nrbnum, nrb, nrb …)` works as you would expect. This form of the call is recommended over the `swait(-1)` form.

After control is returned, the calling program must determine which of the NRBs in the list has completed. This can be done by examining the NRBSTAT field of each of the NRBs.

See “ntxteat.c” and “ntxtgen.c” in the “tests” subdirectory of your NetEx installed distribution for a sample server/client multi-threaded application. It also contains samples of all 3 forms of the ‘swait’ call. The sample of the swait(-1) call is not foolproof!

C SWAIT Examples

Figure 12 shows an example of an SWAIT (0).

```
(program)
.
.
swrit (nrba,buffa,len,dmode);
swrit (nrbb,buffb,len,dmode);
i=0;
while (i++<5&&
      nrba.nrbstat<0&&
      nrbb.nrbstat<0) {

      delay(1);
      swait(0);
}
if (nrba.nrbstat>=0) {
   /*process nrba */
}
else {
   /*process nrbb */
}
```

Figure 12. SWAIT(0) Example

Figure 13 shows an example of an SWAIT(-1).

```
(program)
.
.
swrit (nrba,buffa,len,dmode);
swrit (nrbb,buffb,len,dmode);
swait(-1);
if (nrba.nrbstat>=0) {
   /*process nrba */
}
else {
   /*process nrbb */
}
```

Figure 13. SWAIT(-1) Example

SWAIT Function Format

The SWAIT function has the following format:

Function	**Required Parameters**
`swait`	`(nrbnum, nrb, nrb, ...)`

SWAIT Parameters

The following parameters are used in the SWAIT function. The parameters must be specified in the order presented. All parameters are required.

swait

This is the verb for this function.

nrbnum

This required parameter is the number of NRBs to wait for. Control will be returned after the completion of any calls/NRBs specified. If 0 is specified, the NetEx subroutine library will take control and update NRBs or perform other functions.

nrb

This required parameter is a pointer to one or more NRBs (the number of NRBs specified in nrbnum) associated with the request to be waited for. An *nrb* is required for each NRB specified in *nrbnum*.

SDISC C Function

The SDISC (disconnect) function provides the means for any connected program to terminate a session. The request is immediate and any data currently in transport may not be delivered. If data delivery is required, the program must wait for confirmation of previous SREAD or SWRIT functions before issuing the SDISC.

When an SDISC is issued, an outstanding SREAD in the other program will terminate with an error in NRBSTAT.

NetEx does not ensure that data written with an SDISC macro will actually be received by the other program.

SDISC Function Format

The SDISC function has the following format

Function (Select One)	**Required Parameters**
`sdisc` `sdiscw`	`(nrb, buffer, length, datamd)`

SDISC Parameters

The following parameters are used in the SDISC function. The parameters must be specified in the order presented. All parameters are required.

sdisc
sdiscw

This is the verb for this function. SDISCW specifies that the calling program must wait for the call to complete before processing is resumed.

nrb

This required parameter is a pointer to the NRB data area to be passed to NetEx.

buffer

This required parameter is a pointer to the buffer data area that holds the user data to be sent to the corresponding application.

Note: For SDISC, delivery of DISCONNECT data is **not** reliable, although the actual disconnection will always occur.

length

This required parameter is the length of the data (in addressable units) to be sent to the corresponding application, to be presented with the completion of the corresponding application's SREAD request. If no data needs to be sent to the other application, the length may be set to zero. The data type of *length* should be INT.

datamd

This required parameter is the datamode to be used to send the disconnect data to the corresponding application. The data type of *datamd* should be INT. Refer to NRBDMODE in Chapter 4: NetEx Request Block for a discussion of the datamode parameter.

On completion of the SDISC, the connection will no longer exist; new commands against that connection will be rejected. An SOFFR or SCONN must be issued to establish a new connection.

SDISC Entry Parameters

The following NRB fields are used by SDISC on entry.

NRBBUFA	Address of outgoing Pdata
NRBLEN	Length of outgoing Pdata
NRBUBIT	Pdata unused bit count
NRBDMODE	Datamode of Pdata
NRBPROTA	Address of outgoing Odata
NRBPROTL	Length of outgoing Odata

SDISC Results

The following NRB fields are updated when SDISC completes.

NRBSTAT	Success/failure code
NRBIND	Set to zero

C Program Examples

The following figures are examples of C language offer and connect programs designed for the transfer of computer generated data.

User your ‘C’ compiler to compile these programs specifying the NetEx library.

```
cc –o nsef001 nsef001.c –lntx
cc –o nsef002 nsef002.c –lntx
```

To run the programs, you must specify the hostname with nsef002 and an optional datamode with nsef001 and nsef002.

datamode

The default is 0, if the datamode is not specified.

Figure 14 shows the offering side (nsef001) and Figure 15 shows the connecting side (nsef002) of a NetEx example.

```
/*
 * This is the offering side of the test, the basic sequence is
 *                 offer with data verification
 *                 confirm
 *                 read and write to remote with data verification
 *                 read disconnect
 * the session is terminated if there are any errors.
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netex/netex.h>
#include <netex/error.h>

#define ELEMENTS 500

nrb n;
short buf[ELEMENTS],verf[ELEMENTS];
int step,i,mode, bytes = sizeof(buf[0]);
char *job[] = { "offer","connect","confirm","read","write","disc","wait"};

/*
 * terminate the session on error condition
 * verify > 0 indicates a data verification error.
 */
void
terminate(int verify)
{
        if(verify--)
                printf(" data miscompared at n=%d, buf=%d, verf=%d\n",
                           verify,buf[verify],verf[verify]);
        printf(" **%s** nrbstat=%ld, nrbind=%ld, nrblen=%ld, mode=%ld\n",
                   job[step],n.nrbstat,n.nrbind,n.nrblen,n.nrbdmode);
        sdiscw(&n,(char *)buf,1,mode);
        if(n.nrbstat != SUCCESS || n.nrbind != INDDISCONNECT)
                printf("**disc stat = %ld**, nrbind = %ld, nrblen = %ld\n",
                           n.nrbstat,n.nrbind,n.nrblen);
        exit(-1);
}

int
main(int argc,char *argv[]) /* nsef001 [datamode] */
{
        memset(&n,'\0',sizeof(n));
        memset(buf,'\0',sizeof(buf));
        step = 0;
        printf(" this is the start of nsef001\n");
        if(argc < 2)
                mode = 0;
        else
                sscanf(argv[1],"%x",&mode);
        verf[0] = 1234;

        /* offer, check for success, verify data */
        soffrw(&n,(char *)buf,4,300,"NSEF0011");
        if(n.nrbstat != SUCCESS || n.nrbind != INDCONNECT)
                terminate(0);
        if(buf[0] != verf[0])
                terminate(1);
        buf[0] = 1234;
        buf[1] = 5678;

        /* confirm the connection */
        step = 2;
        sconfw(&n,(char *)buf,8,mode);
        if(n.nrbstat != SUCCESS)
                terminate(0);

        /* read and verify data */
        step = 3;
        sreadw(&n,(char *)buf,400*bytes,120);
        for(i=0; i<400; i++) {
                verf[i] = i+1;
                if(buf[i] != verf[i])
                        terminate(i+1);
        }
        if(n.nrbstat != SUCCESS || n.nrbind != INDDATA || n.nrblen != 400*bytes)
                terminate(0);
        printf(" first read in nsef001 has completed and verified data\n");
        for(i=0; i<420; i++)
                buf[i] = i+1;

        /* write to remote */
        step = 4;
        swritw(&n,(char *)buf,420*bytes,mode);
        if(n.nrbstat != SUCCESS)
                terminate(0);

        /* read and verify data */
        step = 3;
        sreadw(&n,(char *)buf,40*bytes,120);
        for(i=0; i<40; i++) {
                if(buf[i] != verf[i])
                        terminate(i+1);
        }
        if(n.nrbstat != SUCCESS || n.nrbind != INDDATA  || n.nrblen != 40*bytes)
                terminate(0);
        printf(" second read in nsef001 has completed and verified data\n");
        for(i=0; i<420; i++)
                buf[i] = i+1;

        /* write to remote */
        step = 4;
        swritw(&n,(char *)buf,420*bytes,mode);
        if(n.nrbstat != SUCCESS)
                terminate(0);

        /* read the disconnect */
        step = 3;
        sreadw(&n,(char *)buf,1,30);
        /* verify disconnect */
        if(n.nrbstat != SUCCESS ||n.nrbind != INDDISCONNECT)
                terminate(0);
        printf(" reading disconnect in nsef001 completed\n");
        printf(" this test completed successfully nsef001\n");
        exit(0);
}
```

Figure 14. Example: Offering Side of a NetEx Session (nsef001.c)

```
/*
 * this is the connecting side of the test the basic sequence is
 *              connect to remote
 *              read confirm with data verification
 *              write to and read from remote host with data verification
 *              disconnect
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netex/netex.h>
#include <netex/error.h>

#define ELEMENTS 500

nrb n;
short buf[ELEMENTS],verf[ELEMENTS];
int step,i,mode,bytes = sizeof(buf[0]);
char *job[] = { "offer","connect","confirm","read","write","disc","wait"};
char host[sizeof(n.nrbhost) + 1];

/*
 * terminate the session because of an error
 * verify > 0 indicates data verification error
 */
void
terminate(int verify)
{
        if(verify--)
                printf(" data miscompared at n=%d, buf=%d, verf=%d\n",
                       verify,buf[verify],verf[verify]);
        printf(" **%s** nrbstat=%ld, nrbind=%ld, nrblen=%ld, mode=%ld\n",
               job[step],n.nrbstat,n.nrbind,n.nrblen,n.nrbdmode);
        sdiscw(&n,(char *)buf,1,mode);
        if(n.nrbstat != SUCCESS || n.nrbind != INDDISCONNECT)
                printf("**disc stat = %ld**, nrbind = %ld, nrblen = %ld\n",
                       n.nrbstat,n.nrbind,n.nrblen);
        exit(-1);
}

int
main(int argc,char *argv[]) /* nsef002 host [datamode] */
{
        memset(&n,'\0',sizeof(n));      /* zero out nrb */
        memset(buf,'\0',sizeof(buf));
        printf(" this is the start of nsef002\n");
        if(argc == 2)
                mode = 0;
        else if(argc == 3)
                sscanf(argv[2],"%x",&mode);
        else {
                printf(" nsef002: wrong arg count\n");
                exit(1);
        }
        strncpy(host,argv[1], sizeof(host));
        verf[0] = 1234;
        verf[1] = 5678;
        buf[0] = 1234;

        /* connect to remote */
        step = 1;
        sconnw(&n,(char *)buf,4,mode,"NSEF0011",host);
        if(n.nrbstat != SUCCESS)
                terminate(0);
        step = 3;

        /* read confirm, verify data */
        sreadw(&n,(char *)buf,8,120);
        for(i=0; i<2; i++)
                if(buf[i] != verf[i])
                        terminate(i+1);
        if(n.nrbstat != SUCCESS || n.nrbind != INDCONFIRM)
                terminate(0);
        printf(" reading the sconfw completed successfully");
        for(i=0; i<400; i++)
                buf[i] = i+1;

        /* write to remote */
        step = 4;
        printf(" this is the mode before write %d, nrbmode = %ld\n",
               mode, n.nrbdmode);
        swritw(&n,(char *)buf,400*bytes,mode);
        printf(" this is the mode after swrite %d, nrbmode = %ld\n",
               mode, n.nrbdmode);
        if(n.nrbstat != SUCCESS)
                terminate(0);

        /* read from remote, verify data */
        step = 3;
        sreadw(&n,(char *)buf,420*bytes,30);
        for(i=0; i<420; i++)
                if(buf[i] != (verf[i] = i+1))
                        terminate(i+1);
        if(n.nrbstat != SUCCESS || n.nrbind != INDDATA || n.nrblen != 420*bytes)
                terminate(0);
        printf(" second read in nsef002 completed with data verified\n");

        /* write to remote */
        step = 4;
        for(i=0; i<40; i++)
                buf[i] = i+1;
        swritw(&n,(char *)buf,40*bytes,mode);
        if(n.nrbstat != SUCCESS)
                terminate(0);

        /* read from remote, verify data */
        step = 3;
        sreadw(&n,(char *)buf,420*bytes,120);
        for(i=0; i<420; i++)
                if(buf[i] != verf[i])
                        terminate(i+1);
        if(n.nrbstat != SUCCESS || n.nrbind != INDDATA || n.nrblen != 420*bytes)
```

```
                terminate(0);
        printf(" third read in nsef002 completed with data verified\n");

        /* disconnect */
        step = 5;
        sdiscw(&n,(char *)buf,1,0);
        if(n.nrbstat != SUCCESS || n.nrbind != 0) {
                printf(" ***disconnect*** nrbstat=%ld, nrbind=%ld\n",
                           n.nrbstat,n.nrbind);
                exit(-1);
        }
        printf(" this test completed successfully nsef002\n");
        exit(0);
}
```

Figure 15. Example: Connecting Side of a NetEx Session (nsef002.c)

Chapter 6: Operator Interface

The NetEx operator interface (NTXOPER) provides a set of commands that allow you to control and display NetEx resources. This chapter details the following information on NTXOPER.

- Executing Operator commands
- Enabling the Remote Operator interface
- Operator command descriptions

Executing Commands

You can execute NetEx operator commands in either command line or interactive mode.

Any characters following the command and parameter entries will be ignored

Command Descriptions Conventions

The following notational conventions are used in the Operator Command Help functions.

Format	Description
`{ }`	One of the selections within the brackets are required.
`[ ]`	One of the selections within the brackets are optional
< >	A required user input field
|	Separator for the selections within brackets.

Command Line Mode via Console

When you execute operator commands in command line mode, you initiate the NTXOPER program for each command and provide the command parameters as arguments to the program. To execute an operator command in command line mode, use the following general format:

```
keyin operator_command parameters
```

Replace `operator_command` with a valid operator command; replace `parameters` with the parameters for the operator command. For example, to execute the DISPLAY HOST command when keyin is configured to ntxd, enter the following:

```
ntxd display host
```

You can also execute an operator command in command line mode on a remote host using the following general format:

```
ntxd : host_name  operator_command  parameters
```

Replace `host_name` with the name of the host on which you want to execute the command; replace `operator_command` with a valid operator command; replace `parameters` with parameters for the command. For more information on executing operator commands on a remote host, refer to Command Descriptions later in this chapter.

Interactive Mode via Telnet Session

When you execute operator commands interactively, you first start an operator interface session using the @xqt *netex*ntxd-exec*.ntxoper command; where *netex*ntxd-exec* follows your site naming convention. During an NTXOPER session, all valid operator commands are available. The following table show the interactive operator command menu:

Interactive Operator Command	**Description**
`. (period)`	Repeats the previous command.
`<operator_command>`	Execute the specified local operator command.
`help | ?`	Show local operator commands
`{ help | ? } <operator_command>`	Show local operator command syntax
`lhelp`	Display the help for the interactive NTXOPER program.
`{ >| rem | rmt | : } hostname operator_command`	Executes the specified operator command using the remote NTXOPER interface.
`q | quit | exit | bye | stop`	Terminates NTXOPER.
`TERM`	A TERM command entered with a system Keyin will terminate the NetEx program running on the system. A TERM entered from a telnet session will terminate the ntxoper program running in the terminal window.

Enabling the Remote Operator Service

The NetEx remote operator service allows you to request a NetEx operator display from other hosts on the network. You can request the display from any NetEx that has the remote operator display feature enabled.

To enable the remote operator service, use the SET NTXOPER operator command. For example, to enable remote operator service on your local host, enter:

```
ntxd set ntxoper on
```

You can define the class of commands available to remote operators with the SET ROPCLASS operator command. For example, to allow remote operators to execute all commands, enter the following:

```
ntxd set ropclass a
```

To allow remote operators to use only general display commands, enter the following:

```
ntxd set ropclass g
```

When the remote operator service is enabled, you can execute NTXOPER operator commands on a remote system either in command line or interactive mode (refer to Executing Commands earlier in this chapter). Any display you request is shown in the format defined by the remote program. For example, if you request a display from a NetEx for zOS, use a NetEx for zOS command and the system returns a NetEx for zOS display.

Operator Command Descriptions

This section details all the NetEx operator interface commands. The following table briefly summarizes each command.

Operator Command	**Description**
CLEAR IPROUTE	Clear IP routing table entries
DISPLAY DELETE	Displays NetEx control blocks which are pending deletion
DISPLAY DRAINED	Displays drained adapters
DISPLAY HOST	Displays host(s) defined to NetEx
DISPLAY IPROUTE	Displays routing table entries
DISPLAY KEY	Displays the current license key
DISPLAY MEMORY	Displays the current memory allocations for various internal control blocks and data buffers. (Same as the DISPLAY MEMORY command.)
DISPLAY NETWORK	Lists network connections currently pending or in progress
DISPLAY PARMS	Displays parameter values controlled by SET commands
DISPLAY SESSION	Lists sessions currently pending or in progress
DISPLAY TRANSPORT	Lists the current state of transport connections
DISPLAY VERSION	Displays the current version level of the NetEx/IP component.
DRAIN ADAPTER	Prevents new connections from using the adapter
DRAIN HOST	Prevents new connections with a specific host
DRAIN NETEX	Prevents any new connections from starting
DRAIN PATH	Prevents any new connection from using the path between specified GNAs
HALT SREF	Immediately stops a NetEx session
HELP	Provide a list of commands or details the command syntax
LHELP	Provide help information in interactive mode
KILL NETEX	Immediately stops all NetEx resources
LOAD KEY	Loads the current license key
LOAD NCT	Transfers a pamfile created by the Configuration Manager to NetEx
SET BUFOLIM	Specifies the maximum buffers (segments) that may be outstanding
SET CONTIME	Specifies the connect attempt timeout value
SET DBGDATA	Specifies number of data bytes to trace
SET DBGMSG	Enable/disable NetEx message tracing
SET DBGREQ	Enable/disable user request tracing

Operator Command	**Description**
SET DBGRET	Enable/disable user response tracing
SET DEADTIME	Specifies time to wait until disconnecting a connection due to lack of traffic
SET DEFBI	Specifies the default maximum input buffer size for sessions
SET DEFBO	Specifies the default maximum output buffer size for sessions
SET IDLETIME	Specifies time between idle messages to verify the continued existence of the other end of the connection
SET IPROUTE	Set an IP routing table address entry
SET MAXBI	Specifies the maximum input buffer size a user may request on a SCONN or SOFFR
SET MAXBO	Specifies the maximum output buffer size a user may request on a SCONN or SOFFR
SET MAXKBS	Specifies the maximum rate at which NetEx/IP will deliver data to the network for each network connection
SET MBUFIN	Specifies the maximum number of buffers available for incoming data
SET MBUFOUT	Specifies the maximum number of buffers available for outgoing data
SET MSGLVL	Specifies the level of messages to display
SET MSGSYSLOG	Specifies where NetEx messages are logged
SET MULTIHOST	Set multihost option on or off
SET NTXOPER	Enables and disables the remote operator service
SET POLLSEL	Specifies whether to poll with a timeout or not
SET PREFPROT	Specifies the default preferred protocol type to use when connecting to hosts that support multiple NetEx/IP protocol types. *(Although Protocol 4 was introduced in Release 6, it is not yet validated on Unisys.)*
SET RCVDATAQHB	Set high threshold (in bytes) for each connection data queue (Prot4 only) - subsequent blocks are NAK'ed until data queue is to rcvdataqlb
SET RCVDATAQHS	Set high threshold (in segments) for each connection data queue (Prot4 only) - subsequent blocks are NAK'ed until data queue is to rcvdataqls
SET RCVDATAQLB	Set low threshold (in bytes) for each connection data queue (see SET RCVDATAQHB) – (Prot4 only)
SET RCVDATAQLS	Set low threshold (in segments) for each connection data queue (see SET RCVDATAQHS) – (Prot4 only)
SET READTIME	Time that NetEx will retain incoming data waiting for a user read request
SET REXMWBLKS	If userexmitq is being used set number of blocks to calculate wait time before retransmitting a block that was NAK'ed
SET ROPCLASS	Class of operator commands that the remote operator may issue

Operator Command	**Description**
SET SESMAX	Number of concurrent session offers and active connections allowed
SET USERCVGAPQ	Specifies whether or not to use the receive gap queue
SET USEREXMITQ	Specifies whether or not to use the retransmit queue
SET WDOGINT	Time between NetEx internal checks for timed events
SET XDBG	Specifies whether or not to set detailed tracing of throttling information
START ADAPTER	Reverses the effect of DRAIN ADAPTER
START NETEX	Reverses the effect of DRAIN NETEX
START HOST	Reverses the effect of DRAIN HOST
START PATH	Starts a path which had been drained
SWITCH	Saves the current ‘ntxlog’ file as ‘ntxlog.n’ (where *n* is an integer greater than or equal to 1), and starts using a new ‘ntxlog’ file.
SWLOG	Saves the current ‘ntxlog’ file as ‘ntxlog.n’ (where *n* is an integer greater than or equal to 1), and starts using a new ‘ntxlog’ file.

Figure 16. NetEx Operator Commands

CLEAR IPROUTE

Description

Clear all or a selected IP routing table entry. Use caution using the “`clear ipr all`” – it is useful only if you plan to do a “`load nct`” right afterwards or to manually re-enter all entries.

Format

```
Clear  IProute  {<GNA> | ALL}
```

Parameters

GNA

The NCT defined NETADDR and SMGDREF of a host adapter.

ALL

Clear all table entries

DISPLAY DELETE

Description

Displays any NetEx defined control blocks pending deletion. This is typically only useful for NESi support for troubleshooting.

Format

```
Display  DELETE
```

DISPLAY DRAINED

Description

The operator may display a list of drained adapters.

Format

```
Display DRAINED
```

Display Examples

The following figure shows a sample DISPLAY DRAINED:

```
NtxOper:
>d drained
PROD$CONF=NETEX*NTXD-CFG.
11:23:18         Host UNID41C        Drained Adapters
GNA   Source   Location   GNA   Source   Location
----  -------  --------   ----  -------  --------
e101  OP CMD   RMTADAPT
```

Figure 17. Display Drained Adapter Command Output.

The columns in the display are as follows:

GNA

The GNA of the drained adapter.

Source

How the adapter became drained. The following indicate the possible ways the adapter can be drained:

- OP CMD – via an NTXOper command
- I/O ERR – critical error on the network

Location

This column indicates whether the adapter drained is local or remote:

- LCLADAPT – local
- RMTADAPT - remote

DISPLAY HOST

Description

Lists the hosts defined on the network. By specifying a host name, you can limit the display to only the specified host and receive more detailed information for that host.

Format

```
Display Host [<hostname>]
```

Parameters

hostname

Specifies the name of the host to be displayed. By default, information on all hosts is displayed.

Display Examples

The following figure shows a sample DISPLAY HOST display when the hostname parameter is omitted:

```
NtxOper:
>d h
PROD$CONF=NETEX*NTXD-CFG.
11:27:44         Host UNID41C        Current Paths
Host UNID41C has paths to the following NTX groups
NtxHost   #Hosts  DrainedHosts  State
--------  ------  ------------  -------
AIX          1          0
LINUX        3          0
SOLARIS      5          0
TANLIN       2          0
TANZOS       3          0
UNIFCI       2          0
UNISYS       4          0
ZOS          2          0

Host UNID41C has paths to the following NTX hosts
NTXHost   #Paths  DrainedPaths  State
--------  ------  ------------  -------
AIX3         4          0
ALT1         2          0
ALT2         2          0
BULLZOST     2          0
DOWNEY       2          0
HPA          2          0
IIC          2          0
INTGDOWN     2          0
INTGSUSE     2          0
INTGYELL     2          0
INTGZOSB     2          0
INTGZOSI     2          0
INTGZOST     2          0
LINNG1       2          0
LINYELL      2          0
```

```
LINZOST      2          0
MST_OSS      2          0
MS_OSS       2          0
NTXLCL       4          0
NTXLCL00     1          0
NTXLCL01     1          0
NTXLCL02     1          0
NTXLCL03     1          0
OLDMAN       2          0
OLINUX       2          0
STRATZOS     2          0
SUNBURN      4          0
SUNLIGHT     4          0
SUNRISE      4          0
SUNSET       4          0
SUNSPOT      2          0
SUSE1        2          0
SYBIL        4          0       DRAINED
TANDOWN      2          0
TANIIC       2          0
TANSUSE      2          0
TANYELL      2          0
TANZOSB      2          0
TANZOSI      2          0
TANZOST      2          0
UNID4150     8          0
UNID4152     8          0
UNID41C      2          0
UNID41D      4          0
UNIFCI1      4          0
UNIFCI2      4          0
WSV0332      2          0
WSV0864      2          0
WSV1264      2          0
WSV1264T     2          0
YELLOWST     2          0
YELLZOSI     2          0
YELLZOST     2          0
ZOS5         4          0
ZOSA         2          0
ZOSB         2          0
ZOSI         2          0
ZOSK         2          0
ZOSR         2          0
ZOST         2          0
ZOSY         2          0
```

Figure 18. Display Hosts Command Output, No HostName specified.

The columns in the display are as follows:

NTXhost

The name of each host or group on the network. The names correspond to the names used on the NCT data file HOST statement.

#Hosts

This column is only under the Groups and indicates the number of hosts in the NTXhost group as defined in the NCT data file HOST definitions.

DrainedHosts

This column is only under the Groups and indicates the number of drained NTXhosts in the group. Hosts are drained by NTXOper command DRAIN Host.

#Paths

This column is only under the Hosts and indicates the number of paths defined by the Configuration Manager to reach the NTXhost.

State

If a group/host is currently drained, DRAINED appears for that NTXhost name.

The following display is an example of the DISPLAY HOST command when a host name is supplied:

```
NtxOper:
>d host sybil
PROD$CONF=NETEX*NTXD-CFG.
11:35:17         Host UNID41C        Paths to SYBIL
Host UNID41C has following paths to SYBIL but is DRAINED
SYBIL     Protocol: 2 4                   DRAINED
          Options: None
          NextPath  FrGNA  ToGNA  State
          --------  -----  -----  -------
                    0c30   e101
                    0c30   1901
                    0d30   e101   DRAINED
                    0d30   1901
```

Figure 19. Display Host Command, HostName specified

The fields in the display are as follows:

PROD$CONF

This shows the current NetEx configuration file

Host

This field shows the NetEx name of the local host.

Paths to

This field shows the NetEx name of the destination host

Protocol

This field shows what NetEx protocol is supported.

DRAINED

Only present if the host is DRAINED

Options

This field shows the possible options supported:

NOAPR, ALTFIRST, LONGMSG

Next Path

When ALTFIRST is specified, this column will indicate the next path to be used with an arrow (->).

FrGNA

This column shows the GNA(s) on the local host for each path.

ToGNA

This column shows the GNA(s) on the remote host for each path.

INTRA indicates the path is internal to this host.

State

This column will indicate if a path is DRAINED.

The following display is an example of the DISPLAY HOST command when a group name is supplied:

```
NtxOper:
>d host zos
PROD$CONF=NETEX*NTXD-CFG.
14:01:19         Host UNID41C        Paths to ZOS
Host UNID41C has following paths to ZOS
ZOSI      Protocol: 2 4
          Options: None
          NextPath  FrGNA  ToGNA  State
          --------  -----  -----  -------
                    0c30   0700
                    0d30   0700
ZOST      Protocol: 2 4
          Options: None
          NextPath  FrGNA  ToGNA  State
          --------  -----  -----  -------
                    0c30   0100
                    0d30   0100
```

Figure 20. Display Host Command, GroupName specified

DISPLAY IPROUTE

Description

Display all or a selected IP routing table entry.

Format

```
Display IProute [<GNA>]
```

Parameters

GNA

The 3 or 4 hexidecimal GNA is defined in the NCT as the NETADDR and SMGDREF of a host adapter.

ALL

Display all table entries. ALL is assumed if no GNA is given.

Display Examples

The following figure shows a sample DISPLAY IPROUTE display when the GNA parameter is omitted:

```
NtxOper:
>d iproute
PROD$CONF=NETEX*NTXD-CFG.
14:03:11         Host UNID41C        Current GNA to IP Mapping Table
GNA   IP Address       Source     GNA   IP Address       Source
----  ---------------  -------    ----  ---------------  -------
0100  10.1.5.156       Local      0200  10.1.5.11        Local
0400  10.1.5.152       Local      0700  10.1.5.153       Local
0800  10.1.5.154       Local      0900  10.1.5.155       Local
0a00  10.1.5.12        Local      0b00  10.1.5.157       Local
0c60  10.1.5.24        Local      0c20  10.1.6.26        Local
0c00  10.1.6.18        Local      0c30  10.1.6.19        Local
5102  10.1.1.54        Local      5105  0.0.0.0          Unknown
5201  0.0.0.0          Unknown    5501  10.1.1.63        Local
5901  10.1.1.53        Local      5a01  0.0.0.0          Unknown
6101  10.1.1.61        Local      8f20  0.0.0.0          Unknown
8f00  0.0.0.0          Unknown    9001  10.1.5.123       Local
9101  10.1.6.123       Local      9201  10.1.5.121       Local
9901  10.1.5.98        Local      9a01  10.1.6.98        Local
9e01  10.1.5.125       Local      9f01  10.1.6.125       Local
a301  0.0.0.0          Unknown    a401  10.1.6.225       Local
a601  10.1.6.168       Local      a901  10.1.5.28        Local
ac01  10.1.5.10        Local      d001  10.1.5.23        Local
d501  10.1.5.31        Local      d601  10.1.6.31        Local
e000  172.29.122.94    Local      e101  10.1.5.75        Local
e201  10.1.5.144       Local      e400  10.1.6.17        Local
```

Figure 21. Display IP Route Command

The columns in the display are as follows:

GNA

The network address mapped by this entry. Currently there will always be 4 leading zeros.

IP Address

The IP address associated with this entry or 0.0.0.0 for none.

Source

The source of this entry. Currently it may be Local (for a successful DNS lookup), Oper (for an operator entry), or Unknown (for DNS lookup failed).

DISPLAY KEY

Description

Displays the NetEx license key

Format

```
Display KEy
```

Display Examples

The following figure shows a sample DISPLAY KEY:

```
NtxOper:
>d key
PROD$CONF=NETEX*NTXD-CFG.
14:06:42         Host UNID41C        License Key
ctlcap=40(IP), ctllicflag=00(), ctltnpsmax= 0

Key = DBPZ-YAA6-ABAA-DTP3-DB3D-LA4A
Protocol = 40(IP)
Not Operational Date = 20170130
Expiration Date = 20161231
```

Figure 22. Display Key Command

DISPLAY MEMORY

Description

Displays the current memory allocations for various internal control blocks and data buffers.

Format

```
Display MEMory
```

Display Examples

The following figure shows a sample DISPLAY MEMORY command output:

```
NtxOper:
>d mem
Using COMAPI mode A index=1
PROD$CONF=NETEX*NTXD-CFG.
14:08:30         Host UNID41C        Memory
ctlbufs=      20  ctlmbufs=      4  ctlnrb=       34  ctlpam=        7
ctlmsg=       24  ctlodata=      8  ctlquehead=   14
ctlquesub=     1  ctlquetub=     2  ctlquenub=     2  ctlquedcb=     2
ctlquebcb=    24  ctlquendb=     0  ctlquetdb=     2
```

Figure 23. Display Memory Command

ctlbufs

# of buffers (of length segment size) currently in use

ctlmbufs

# of dynamically allocated buffers (number of mallocs)

ctlnrb

# of nrb control blocks currently allocated

ctlpam

# of pams currently allocated for active sessions

ctlmsg

# of control messages currently allocated

ctlodata

# of odata buffers currently allocated

ctlquehead

# of queue headers currently allocated

ctlquesub

# of session user blocks currently allocated

ctlquetub

# of transport user blocks currently allocated

ctlquenub

# of network user blocks currently allocated

ctlquedcb

# of device blocks currently allocated

ctlquebcb

# of data buffers currently allocated

ctlquendb

# of network data control *blocks currently allocated*

ctlquetdb

# of transport data control blocks currently allocated

DISPLAY NETWORK

Description

Lists the network connections that are currently pending or in progress on the operator's host. By specifying an NREF, the operator may limit the display to only the specified network connection and receive more detailed information.

Format

```
Display Network [<nref> | all]
```

Parameters

Nref

Specifies a NREF for the command. By default, information for all NREFs is displayed.

ALL

Displays the information for all NREFs as though the NREF was specified

Display Examples

The following shows a sample DISPLAY NETWORK command output:

```
NtxOper:
>display network
PROD$CONF=NETEX*NTXT-CFG.
09:26:16         Host UNIFCIC        Active Network Connections
 Nref   User   State  Rnref   Blk In    Blk Out   LGNA  RGNA
----- -------- ------ ----- ---------- ---------- ----  ----
    1 NTXOPER  offer
    2 BFXJST   offer
   37 DAVEG    data     205        445        674 0f70  0100
65535 SESSMGR  offer
```

Figure 24. Display Network Command

The following defines the fields in the display:

Nref

This field shows the unique identifier that distinguishes this network connection from all other active network connections to this NetEx. This reference identifier must be used in operator commands that modify a network connection, and may be used with this command to get detailed information about this network connection.

User

This field shows the name of the process requesting network services.

State

This field shows the current status of the network connection. This is useful for tracking the progress of a connection, particularly for finding hung connections. The possible states are as follows:

clos-s	A close has been sent by the user. No additional data may be sent, but additional data may be received.
conf	CONNECT message received, waiting for the confirm call.
data	Connection completed and user may exchange data.
conn	Connect request issued by user, waiting for confirm.
disc	Disconnect detected, but not yet complete.
offer	Offer has been issued by user, waiting for connect.
smconn	A user is in the process of connecting to a remote session manager. This process is internal to NetEx. The user's connect is in progress.
assign	A user is in the process of being identified as a network user. This state is internal to NetEx. The user's offer or connect is in progress.

Rnref

This field shows the Nref identifier on the remote host.

Blk In

This field shows the number of user messages received during this session.

Blk Out

This field shows the number of user messages transmitted during this session.

LGNA

This field shows the GNA address (dref) the local NetEx host is using for this comnnection.

RGNA

This field shows the GNA address (dref) the remote NetEx host is using for this comnnection.

The following shows a DISPLAY NETWORK command output when an Nref is specified:

```
NtxOper:
>display n 2
PROD$CONF=NETEX*NTXD-CFG.
08:34:42         Host UNIFCIC        Nref     2
Name=   DAVEG     Pid=  9771458949  State=    data
Writes=     9147  Reads=     17336  Nubblki=   32768  Nubblko=   32768
Readto=       54
Logp21=        0  Log9=          0  Log10=         0  Log11=         0

Physical Address Map (PAM)
 pam header --
    len=14 segsize=fffc maxrate=000000 delay=0000
 pam entries --
   pam entry   1->06 01 82 88 0c 30
   pam entry   2->06 01 42 88 0c 30
 end of pam
```

Figure 25. Display Network Command, NREF specified

The following describes the fields in the display:

Nref

This field shows the unique identifier that distinguishes this network connection from all other active network connections to this NetEx. This reference identifier must be used in operator commands that modify a network connection, and may be used with this command to get detailed information about this network connection.

Name

This field shows the name of the process requesting network services.

Pid

Process ID of the application

State

This field shows the current status of the network connection. This is useful for tracking the progress of a connection, particularly for finding hung connections. The possible states are as follows:

clos-s	A close has been sent by the user. No additional data may be sent, but additional data may be received.
conf	CONNECT message received, waiting for the confirm call.
data	Connection completed and user may exchange data.
conn	Connect request issued by user, waiting for confirm.
disc	Disconnect detected, but not yet complete.
offer	Offer has been issued by user, waiting for connect.
smconn	A user is in the process of connecting to a remote session manager. This process is internal to NetEx. The user's connect is in progress.

assign	A user is in the process of being identified as a network user. This state is internal to NetEx. The user’s offer or connect is in progress.

Writes	This field shows the number of user messages transmitted during this session.
Reads	This field shows the number of user messages received during this session.
Nubblki	Maximum input blocksize.
Nubblko	Maximum output blocksize.
Readto	Read timeout.
Logp21	Number of 2 part messages received.
Log9	Number of “1 part message received when first or second part expected”.
Log10	Number of “Unexpected second part of 2 part message”.
Log11	Number of “Received first, expecting second part”.
PAM	Internal NetEx Path Address Map entry used to establish a connection to the remote NTXHost (from NCT).

DISPLAY PARMS

Description

Displays most parameter values controlled by the SET command and initialization parameters.

Format

```
Display Parms
```

Display Examples

The following figure shows a sample DISPLAY PARMS command output:

```
NtxOper:
>display parm
PROD$CONF=NETEX*NTXD-CFG.
09:09:29         Host UNID41C        Parameters
contime=    30  deadtime=  60  idletime=        6  readtime=       30
maxbi=   65400  maxbo=  65400  defbi=       32768  defbo=       32768
segsize= 32768  wdogint=    2  msglvl=    blither  mtudisc=         0
max ses=    32  max tran=   0  max net=         0  dreadque=       12
cur ses=     2  cur tran=   3  cur net=         3  status=     NORMAL
lim ses=   255  rmtoper=   ON  ropclass=        G  multihost=     OFF
prefprot=    2  maxkbs=     0  xdbg=            0  pollsel=        ON
userexmitq=  1  rexmwblks=  2  bufolim=      2000  usercvgapq=      0
dtqhs=   15000  dtqls=   8000  dtqh=     20000000  dtql=     10000000
dbgmsg=      0  dbgdata=    0  dbgreq=          0  dbgret=          0
NCTVer=    196  mbufin=     8  mbufout=         8
xmem alloc= 30  in_use=     1  failed=          0  comapibuff=    100
msgsyslog=   1
```

Figure 26. Display Parms Command

The following describes the fields:

contime

This field shows the maximum number of seconds that NetEx will wait for a transport connect message to generate a response from the remote host. This parameter may be changed using the SET CONTIME operator command or the CONTIME initialization statement.

deadtime

This field shows the maximum number of seconds that NetEx will wait before it disconnects a transport connection (or attempts an alternate path) because there was no response from a remote host. This parameter may be changed using the SET DEADTIME operator command or the DEADTIME initialization statement.

idletime

This field shows the maximum number of seconds that NetEx will wait before sending an idle message to verify the continued existence of a party at the other end of a logical connection. This parameter may be changed using the SET IDLETIME operator command or the IDLETIME initialization statement.

readtime

Shows the number of seconds that NetEx transport retains user data while waiting for the receiver to issue a read request. This parameter may be changed using the SET READTIME command or the DATATO initialization statement.

maxbi

This field shows the maximum buffer input size (in bytes) that a user may specify for data coming in to his host in a single message. This parameter may be changed using the SET MAXBI operator command or the MAXBI initialization statement.

maxbo

This field shows the maximum buffer output size (in bytes) that a user may specify for data leaving his host in a single message. This parameter may be changed using the SET MAXBO operator command or the MAXBO initialization statement.

defbi

This field shows the default buffer input size (in bytes) that a user may specify for data coming in to his host in a single message. This parameter may be changed using the SET DEFBI operator command or the DEFBI initialization statement.

defbo

This field shows the default buffer output size (in bytes) that a user may specify for data leaving his host in a single message. This parameter may be changed using the SET DEFBO operator command or the DEFBO initialization statement.

segsize

This field shows the maximum segment size that will ever be used for any connection from this host. This parameter may be changed using the SEGSIZE initialization statement.

wdogint

This field shows the number of seconds that the watchdog timer waits before checking NRB timeout values. The parameter may be set using the SET WDOGINT operator command or the TIMER initialization statement.

msglvl

This field shows the minimum level of severity necessary to display a message to the operator. This parameter can be set using the SET MSGLVL command or the MSGLVL initialization statement.

mtudisc

0=don’t; set path mtu discovery off, so that IP will fragment if necessary

Non zero=behave as system settings dictate

max ses

This field shows the number of session connections or OFFERs permitted at one time. This parameter may be changed using the SET SESMAX operator command or the SMAX initialization statement. It can never exceed the value specified for the initialization statement SLIM.

max tran

This field shows the number of direct transport connections or OFFERs permitted at one time. This parameter is always 0 because direct transport connections are not supported at this time.

max net

This field shows the number of direct network connections or OFFERs permitted at one time. This parameter is always 0 because direct network connections are not supported at this time.

dreadque

This field shows the number of reads queued up for each driver connection.

cur ses

This field shows the number of session connections in progress or being OFFERed.

cur tran

This field shows the number of transport connections in progress or OFFERed.

cur net

This field shows the number of network connections in progress of being OFFERed.

status

This field shows the current status of NetEx. Status can be one of the following:

DRAINED	A DRAIN command has been issued and no sessions are active.
DRAINING	A DRAIN command has been issued, but some sessions are still active.
NORMAL	All sessions are active.

lim ses

This is the limit on the maximum number of sessions (see “SET SESMAX” and the init parm ‘smax’)

rmtoper

This field shows whether the remote operator service is ON or OFF. The remote operator status may be changed with the SET NTXOPER command.

ropclass

This field shows the privilege class of remote operator service (may be changed by the SET ROPCLASS command). The classes are as follows:

A	indicates that all commands are allowed.
G	indicates that only display commands are allowed.

multihost

This shows whether multihost is enabled or disabled. Multihost is used with TNP. The NetEx/IP Requester hostnames that are specified in the Requester configuration files are used with the NetEx requests. MULTIHost should be set to ON if there are multiple NetEx/IP Requester hosts using this NetEx with TNP, and the Requester hosts are defined in the NCT. MULTIHost can be set to OFF if there is only one Requester host, or if all SOFFERs will have unique names. Typically, SET MULTIHOST should be set to ON in TNP configurations.

prefprot

This field shows the default preferred protocol type to use when connecting to hosts that support multiple NetEx/IP protocol types.

mxkbs

This field shows the maximum rate at which NetEx/IP will deliver data to the network for each network connection. If zero is specified, data will be delivered with no internal throttling. This value is only used for connections to hosts that do not have a 'rate' value specified in the PAM. This value is specified in Kbits per second. For example, a value of 50 means 50Kbs; a value of 50000 means 50Mbs (i.e. 50,000 Kbs).

xdbg

This is a special debug option to see details of throttling. Setting the value > 0 will turn on xdbg tracing, default is 0.

pollsel

This specifies whether or not the dispatcher will poll with 0 timeout or not. Especially useful in order to get good performance with small segments.

userexmitq

Specifies whether or not to use the retransmit queue.

rexmwblks

This is the number of blocks to use to calculate the time to wait before rexmitting a block when it is NAKed. Only used if userexmitq is 1.

bufolim

This is the maximum number of sent segments allowed waiting for acknowledgement

usercvgapq

This indicates whether or not NetEx should use the receive gap queue

dtqhs

This is the high threshold value (in segments) for the size of the receiving NetEx DataQue for each network connection. If the size of the Data Queue exceeds this value, subsequent blocks received on the network that are greater than the highest received LRN are NAK'ed and discarded. Subsequent blocks will continue to be discarded until the size of the DataQue is reduced to the value specified by 'dtqls'. (Prot 4 only).

dtqls

This is the low threshold value (in segments) for the size of the receiving NetEx DataQue for each network connection. See the 'dtqhs' parameter for the description of how this value is used. (Prot 4 only)

dtqh

This is the high threshold value (in bytes) for the size of the receiving NetEx DataQue for each network connection. If the size of the Data Queue exceeds this value, subsequent blocks received on the network that are greater than the highest received LRN are NAK'ed and discarded. Subsequent blocks will continue to be discarded until the size of the DataQue is reduced to the value specified by ‘dtql’. (Prot 4 only).

dtql

This is the low threshold value (in bytes) for the size of the receiving NetEx DataQue for each network connection. See the ‘dtqh’ parameter for the description of how this value is used. (Prot 4 only)

dbgmsg

Indicates if NetEx message tracing is enabled. Set to 0 or 1, default is 0.

dbgdata

This is the number of data bytes being traced

dbgreq

Indicates if user request tracing is enabled. Set to 0 or 1, default is 0.

dbgret

Indicates if user response tracing is enabled. Set to 0 or 1, default is 0.

NCTVer

This is the version of the NCT from which the current pamfile was created**.**

mbufin

This is the maximum number of buffers outgoing, unless overridden by rate, delay, and/or segsize considerations.

mbufout

This is the maximum number of buffers incoming, unless overridden by rate, delay, and/or segsize considerations.

Xmem alloc

The maximum number of buffers allowed for communications between applications and Netex

In_use

The current number of buffers currently in use.

failed

The number of times a buffer was requested and a buffer was not available. The Request is retried.

comapibuff

Specifies the listen queue size for COMAPI. This value cannot exceed 20% of COMAPI MAX_UDP_BUFFERS

msgsyslog

This specifies where NetEx msgs will be output. A value of 1 means output to the NetEx log. A value of 0 means minimal NetEx log output.

DISPLAY SESSION

Description

Lists the sessions that are currently pending or in progress on the operator’s host. By specifying a Sref, you can limit the display to the specified session and receive more detailed information for the session.

Format

```
Display Session [<sref> | all]
```

Parameters

Sref

Specifies the Sref to be displayed. By default, all sessions for all Srefs are displayed.

ALL

Displays the information for all SREFs as though the SREF was specified

Display Examples

The following shows a sample DISPLAY SESSION command output:

```
NtxOper:
>display session
PROD$CONF=NETEX*NTXD-CFG.
08:35:31         Host UNID41C        Active Sessions
 Sref   User    Pid  State    Name     Host   Rnref  Msg In  Msg Out
----- -------- ----- ------ -------- -------- ----- -------- --------
    1 NTXOPER  60325 offer  NTXOPER  UNID41C
    2 DAVEE    58949 data   NTXMEAT  UNID41C      3    19957        1
    3 DAVEG    58885 data   NTXMEAT  UNID41C      2        1    19957
```

Figure 27. Display Session, no SREF specified

The following shows a sample DISPLAY SESSION command output when a Sref is specified:

```
NtxOper:
>d s 2
PROD$CONF=NETEX*NTXD-CFG.
08:35:36         Host UNID41C        Sref     2
User=   DAVEE     State=    data    Dest=   UNID41C   Rnref=         3
Maxbi=     65399  Maxbo=     58136  Class=        2  Rate=          0
Reads=     20521  Writes=        1  Pid=  9771458949  Name=   NTXMEAT
Ruser=            Flag=         00  Tid=    00000000
```

Figure 28. Display Session, SREF specified

The following describes the fields in the display:

Sref

This field shows the unique identifier that distinguishes this session from all other active session connections to this NetEx. This reference identifier must be used in operator commands that modify a session, and may be used with this command to get detailed information about this session.

User

This field shows the username requesting session services.

Pid

This field shows the process id of the user of this session.

State

This field shows the current status of the session connection. This is useful for tracking the progress of a connection, particularly for finding hung connections. The possible states are as follows:

clos-s	A close has been sent by the user. No additional data may be sent, but additional data may be received.
conf	CONNECT message received, waiting for the confirm call.
data	Connection completed and user may exchange data.
conn	Connect request issued by user, waiting for confirm.
disc	Disconnect detected, but not yet complete.
offer	Offer has been issued by user, waiting for connect.
smconn	A user is in the process of connecting to a remote session manager. This process is internal to NetEx. The user's connect is in progress.
assign	A user is in the process of being identified as a network user. This state is internal to NetEx. The user's offer or connect is in progress.

Name This field indicates the offer name put up by the application**.**

Host Indicates the name of the NTXHost that put up an offer or the remote NTXhost name if the session is established**.**

Rnref

This field shows the destination (or remote) host's network reference number for this session connection. If a connection does not currently exist, this column will be blank. If the connection is a type 1 connection, this number represents the destination host's session reference number.

Msg In / Reads

This field shows the number of user messages received during this session.

Msg Out / Writes

This field shows the number of user messages transmitted during this session.

Maxbi

Maximum input buffer for the session

Maxbo

Maximum output buffer for the session

Ruser

Remote User ID; not used in H300e

Flag

Not defined for H300e

Tid

For NTX Requestor applications, this field shows the Task ID in hexadecimal

DISPLAY TRANSPORT

Description

Displays the current state and progress of all transport services requested directly by user processes (not yet implemented) and indirectly by user requests of session services. By specifying a Tref, the operator limits the display to the specified transport and receives detailed information from the Transport User Block (TUB).

Format

```
Display Transport [<tref> | all]
```

Parameters

Tref

Specifies the Tref of a transport service to be displayed. By default, the services for all Trefs are displayed.

ALL

Displays the information for all TREFs as though the TREF was specified

Display Examples

The following shows a sample DISPLAY TRANSPORT command output:

```
NtxOper:
>display transport
PROD$CONF=NETEX*NTXD-CFG.
08:34:54         Host UNID41C        Active Transport Connections
 Tref   User   Segsz State  Rnref   Blk In    Blk Out    Rexmit
----- -------- ----- ------ ----- ---------- ---------- --------
    1 NTXOPER  32768 offer
    2 DAVEE    32768 data       3      18075          1        0
    3 DAVEG    32768 data       2          1      18076        0
65535 SESSMGR  32768 offer
```

Figure 29. Display Transport, no TREF specified

The following show sample DISPLAY TRANSPORT output, with a TREF specified:

```
NtxOper:
>d t 2
PROD$CONF=NETEX*NTXD-CFG.
08:34:59         Host UNID41C        Tref     2
User=   DAVEE     Name=   NTXMEAT   State=    data    RmtNref=       3
AckQ=          0  InQ=           0  OutQ=          0  DataQ=         0
Maxtblok=      8  Mblko=     65399  Mrate=         0  Segsize=   32768
Maxrblok=      8  Mblki=     65399  Reads=     18304  Writs=         1
CurrKBS=       0  CurrMsRTDl=    6  PipeB=    523192  CurrOB=        0
MinRTDMs=      0  MaxRTDMs=      0  LclSnRt=       0  LSRMax=        0
Prot=          2  RcvKBS=      116  RmRcvRt=       0  RRRMax=        0
RateP=       200  EquivP=      750  DtQB=          0  DtQHM=      8439
CurPmRt=       0  MaxNPDU=   32768  NrbMxRt=       0  DtQHMS=        3
BufOut=        0  BufOLim=    2000  BufOLimC=      0  UseRcvGapQ=    0
ReXmitQ=       0  UseReXmQ=      1  ReXmWBlks=     2  RcvGapQ=  0    0
DtQHB=  20000000  DtQLB=  10000000  DtQHS=     15000  DtQLS=      8000
Transmitter:      Olrn=          1  Plrn=          9  Opbn=          1
                  Rexmt=         0  Tack=   00000000  Tpbn=          1
                  Curtb=  022608ff  Tto=           6  Ackcr=         2
Receiver:         Ilrn=      18305  Rplrn=     18311  Rlrn =         0
                  Rdup=          0  Rack=   00000000  Rpbn =     18304
                  Currb=  00293a07  Rto=           6  Ackcr=         2
                  PbnOO=         0
```

Figure 30. Display Transport, TREF specified

The following describes the fields in the displays:

Tref

This is the NetEx transport reference identifier.

User

This is the name of the process requesting transport services.

Name

The Offer/Connect name

State

This is the current status of the connection. This is useful for tracking the progress of a connection, particularly for finding “hung” connections. The possible states are as follows:

clos-s	A close has been sent by the user. No additional data may be sent, but additional data may be received.
conf	CONNECT message received, waiting for the confirm call.
data	Connection completed and user may exchange data.
conn	Connect request issued by user, waiting for confirm.
disc	Disconnect detected, but not yet complete.

offer	Offer has been issued by user, waiting for connect.
smconn	A user is in the process of connecting to a remote session manager. This process is internal to NetEx. The user's connect is in progress.
assign	A user is in the process of being identified as a network user. This state is internal to NetEx. The user's offer or connect is in progress.

Rnref/RmtNref

This is the remote network reference number.

AckQ

This is the number of blocks queued waiting for acknowledgement.

InQ

This is the number of segments waiting to be sent when proceed and throttle allow it.

OutQ

This is the number of segments queued to be sent.

DataQ

This is the number of blocks (segments) queued waiting for read requests.

Maxtblok

This is the maximum number of transmitting buffers.

Mblko

This is the maximum output block size (bytes).

MRate

This is the maximum transmission rate in Kbits/sec.

Segsize

This is the transport layer segmentation size.

Maxrblok

This is the maximum number of receiving buffers.

Mblki

This is the maximum input block size (bytes).

Reads

This is the number of reads completed by transport.

Writs

This is the number of writes issued by transport.

CurrKBS

This is the current target transmission rate in KBytes/Sec

CurrMsRTDl

This is the current millisecond round trip delay.

PipeB

This is the current size of the pipe in bytes.

CurrOB

This is the current number of bytes sent but not acked.

MinRTDms

This is the minimum round trip delay detected.

MaxRTDms

This is the maximum round trip delay detected.

LclSnRt

This is the current local data from user send rate in KBytes/sec.

LSRMax

This is the maximum local data from user send rate in Kbytes/sec.

Prot

This is the protocol in use for this connection.

RcvKBS

This is the current data to user receive rate in KBytes/sec.

RmRcvRt

This is the current remote data to user receive rate in Kbytes/sec. (Prot 4 only).

RRRMax

This is the maximum data to user receive rate in KBytes/sec.

RateP

This is the percentage factor used to increase or decrease the send rate for each network connection. The send rate may be increased or decreased after the expiration of the interval specified by ‘sndrateupsecs’. The value represents a percentage multiplied by a factor of 10. (Prot 4 only)

EquivP

This is the percentage factor used to determine equivalence of the send and receive rates for each network connection. These rates are assumed to be equal if they fall within this percentage of each other. This value can be dynamically adjusted based on activity and performance of the network. The value represents a percentage multiplied by a factor of 10. (Prot 4 only)

DtQB

This is the number of bytes currently queued to be read by the user.

DtQHM

This is the high mark of the number of bytes queued to be read by the user**.**

CurPmRt

This is the rate in the PAM in Kbits/sec.

MaxNPDU

This is the maximum npdu (segment size) which will be sent over the current path .

NrbMxRt

This is the maximum rate in Kbits/sec specified in the user's NRB.

DtQHMS

This is the high mark of the number of segments queued to be read by the user**.**

BufOut

This is the number of segments waiting to be sent when proceed and throttle allow it.

BufOLim

This is the maximum number of sent segments allowed waiting for acknowledgement.

BufOLimC

This is the number of times segments were ready to be sent and were delayed because BufOLim was reached.

UseRcvGapQ

This indicates whether or not NetEx should use the receive gap queue.

ReXmitQ

This is the number of blocks currently on the retransmit queue.

UseReXmQ

This specifies whether or not the retransmit queue should be used.

ReXmWBlks

This is the number of blocks to use to calculate the time to wait before rexmitting a block when it is NAKed. Only used if UseReXmQ is 1.

RcvGapQ

This is the number of blocks currently on the receive gap queue. .

DtQHB

This is the maximum number of bytes allowed on the incoming data queue. (Prot 4 only).

DtQLB

This is the number of bytes that must be reached on the data queue in order to allow more to be queued if the maximum number was reached. (Prot 4 only).

DtQHS

This is the maximum number of segments allowed on the incoming data queue. (Prot 4 only).

DtQLS

This is the number of segments that must be reached on the data queue in order to allow more to be queued if the maximum number was reached. (Prot 4 only).

Transmitter:

Olrn

This is the last LRN (Logical Record Number) assigned

Plrn

This is the last proceed LRN received from the remote. (Prot 2 only).

Opbn

This is the last PBN (Physical Block Number) assigned.

Rexmt

This is the number of blocks retransmitted.

Tack

This is the ACK/NAK information, bit signal received.

Tpbn

This is the highest PBN the remote side has received – this is the PBN associated with Tack.

Curtb

This is the memory location of the current outgoing data buffer.

Tto

This is the transmit timeout (idle time).

Ackcr

This is the outgoing ACK credit (number of messages before an idle ACK).

Receiver:

Ilrn

This is the next LRN to be given to the user.

Rplrn

This is the most recent proceed LRN sent. (Prot 2 only).

Rlrn

This is the LRN of the top block on the incoming data queue.

Rdup

This is the number of duplicate LRNs received.

Rack

This is the ACK/NAK information, bit signal to be sent.

Rpbn

This is the highest PBN received – this is the PBN associated with Rack..

Currb

This is the memory location of the current receive data buffer.

Rto

This is the receive timeout (communications lost).

Ackcr

This is the incoming ACK credit.

PbnOO

This is the number of blocks (PBNs) received out of order.

DISPLAY VERSION

Description

Displays the current version level of the NetEx/IP component.

Format

```
Display Version
```

Display Example

The following shows the output from a DISPLAY VERSION command for H300e:

```
NtxOper:
>display version
PROD$CONF=NETEX*NTXD-CFG.
14:48:32         Host UNID41C        Version
H300E  Rel 7.0.5-9322
```

Figure 31. Display Version Command

DRAIN ADAPTER

Description

When a local adapter is drained, NetEx/IP immediately stops writing to or reading from the drained adapter. This is true regardless of the state of connections using that adapter. If a remote host attempts to establish a new connection on a route through a drained adapter, the local NetEx/IP will ignore those messages sent by the remote host.

Non-local adapters are adapters that are not attached to the host on which this NetEx/IP is running. After non-local adapters are drained, all new connections are established using routes through other adapters.

Once a session connection is established through a remote adapter that session continues to completion even after a DRAIN command has been entered for it.

Format

```
DRAIN Adapter <GNA>
```

Parameters

GNA

The four HEX character GNA (DREF) of the adapter to be drained.

DRAIN NETEX

Description

Prevents new sessions from being established. This command gracefully terminates all NetEx activity. When all sessions have completed, a message appears indicating that NetEx is drained. NetEx will await a START command to resume normal operation.

Connections in progress are not affected. Offers are taken down with a 3522. Local users attempting to establish a connection receive a 3505 NRBSTAT code. Remote users attempting to establish a connection receive a 3523 NRBSTAT code.

Format

```
DRAIN NETEX
```

DRAIN HOST

Description

Prevents users from establishing a connection with a host. When a user attempts to establish a connection, they receive a 3507 NRBSTAT code.

Format

```
DRAIN HOST <hostname>
```

Parameters

hostname

Specifies the remote host to be drained.

DRAIN PATH

Description

Prevents users from establishing a connection with a host via a specific path. If there are no available paths when a user attempts to establish a connection, they receive a 3508 NRBSTAT code.

Format

```
DRAIN PATH <from-gna> <to-gna>
```

Parameters

from-gna

Specifies the local GNA of the path to be drained.

to-gna

Specifies the remote GNA of the path to be drained.

HALT SREF

Description

Immediately terminates a session. Local users with an active read receive a 3422 NRBSTAT code or a 3100 code on the next write request. An outstanding OFFER is terminated with a 3422 NRBSTAT code.

Format

```
Halt SRef <sref>
```

Parameters

sref

Specifies the SREF for the session to be halted. To determine the SREF number, use the "DISPLAY SESSION" command.

HELP and ?

Description

HELP provides a list of NetEx/IP operator commands and command formats or command syntax.

? provides a list of NetEx/IP operator commands and command formats or command syntax.

Format

```
HElp
?
HElp <operator_command>
? <operator_command>
```

Parameters

operator_command

Optionally, one of the possible NetEx/IP operator commands.

Comments

This command can be entered as either 'help' or '?' for NetEx/IP operator commands. If a valid NetEx/IP operator command is specified the command syntax will be displayed.

LHELP

Description

LHELP provides a list of interactive mode commands and command formats.

Format

LHELP

Parameters

None.

Comments

Refer to HELP for the list and syntax of further NetEx/IP operator commands.

KILL NETEX

Description

Immediately stops NetEx resources and terminates all NetEx activity. Connections in progress are terminated and a 0512 return code is inserted into all active NRBs.

Format

```
KILL NETEX
```

Parameters

None.

LOAD KEY

Description

This command loads the license key located in the NESikeys file defined in the CONFIG.prodconf file

Format

```
Load KEy
```

Parameters

None.

LOAD NCT

Description

Transfers PAM files (data structures describing paths to remote hosts) created by the configuration manager to NetEx. Local adapter information cannot be changed by using this command.

The operator must first modify the NCT data file containing the configuration statements and run the CM utility to create the PAM file (refer to Step 3 Build an NCT on page 151). The filename given on the MAKEPAM statement must be the same name given on the LOAD command. Also, the local hostname specified on the MAKEPAM statement must be the same as the local host in NetEx (the local parameter in the NTXDEFAULT file). If the message “file not in PAMFILE format” is returned, check to see if your host name matches the NetEx local hostname.

Format

```
Load NCt <filename>
```

Parameters

filename

Specifies the fully qualified name of the PAM file to be loaded.

SET BUFOLIM

Description

Specifies the maximum number of buffers (segments) that may be outstanding at any time for a session. Prot 4 only.

Format

```
Set BUFOlim <number-of-buffers>
```

Parameters

number-of-buffers

Specifies the number of buffers (segments). The default is 2000. (Range is 1-2147483647)

SET CONTIME

Description

Specifies the maximum number of seconds that NetEx will wait for a transport connect message to generate a response from the destination host. If this time is exceeded, the transport will assume the destination host is down and return appropriate status to the user. The transport connect message is resent every IDLETIME seconds until CONTIME seconds have passed.

Format

```
Set Contime <number-of-seconds>
```

Parameters

number-of-seconds

Specifies the number of seconds that NetEx waits to generate a response to a transport connection message. The default is 30. (Range is 1-999)

SET DBGDATA

Description

Specifies the maximum number of data bytes to trace for any associated data block. To see the actual data, also set MSGLVL to “blither”.

Format

```
Set DBGDATA <number-of-bytes>
```

Parameters

number-of-bytes

Specifies an integer to indicate the maximum number of data bytes to trace for any associated data block. The default is 0, which indicates that debug tracing is disabled. (Range is 0-2147483647)

SET DBGMSG

Description

Enables or disables the tracing of messages between NetEx and the network. To see the actual data, also set msglvl to blither.

Format

```
Set DBGMSG <value>
```

Parameters

value

Specifies whether debug tracing is enabled or disabled. Specify 0 to disable tracing; specify any other value to enable tracing. The default is 0.

SET DBGREQ

Description

Enables or disables the tracing of user requests arriving at the NetEx protocol stack. When tracing is enabled, the user’s NRB is traced. To see the actual data, also set msglvl to blither.

Format

```
Set DBGREQ <value>
```

Parameters

value

Specifies a value to indicate whether user requests are traced. Specify 0 to disable tracing; specify any other value to enable tracing. The default is 0.

SET DBGRET

Description

Enables or disables the tracing of user responses returned from the NetEx protocol stack. When tracing is enabled, the state of the user’s final NRB is traced. To see the actual data, also set msglvl to blither.

Format

```
Set DBGRET <value>
```

Parameters

value

Specifies a value to indicate whether user responses are traced. Specify 0 to disable tracing; specify any other value to enable tracing. The default is 0.

SET DEADTIME

Description

Specifies the amount of time transport waits until it disconnects a connection because there was no response from the remote host. The remote host normally generates an idle message every IDLETIME seconds based on its own IDLETIME parameter. Receipt of any message from the remote host keeps the DEADTIME timer from expiring.

Format

```
Set DEadtime <number-of-seconds>
```

Parameters

number-of-seconds

Specifies the number of seconds (1-999) that NetEx waits until it disconnects a connection due to no response from the remote host. The default is 60.

SET DEFBI

Description

Specifies the default input buffer size for a connection. This default value is used if the user does not specify a maximum input buffer size in the CONNECT or OFFER request.

Format

```
Set DEFBI <number-of-bytes>
```

Parameters

number-of-bytes

Specifies the default input buffer size in bytes. DEFBI can be from 2048 bytes to the MAXBI value. The default is 32768.

SET DEFBO

Description

Specifies the default output buffer size for a connection. This default value is used if the user does not specify a maximum output buffer size in the CONNECT or OFFER request.

Format

```
Set DEFBO <number-of-bytes>
```

Parameters

number-of-bytes

Specifies the default output buffer size in bytes. DEFBO can be from 2048 bytes to the MAXBO value. The default is 32768.

SET IDLETIME

Description

Specifies the amount of time that transport will wait before sending an idle message to verify the continued existence of a party at the other end of a logical connection. The transmission of any message resets the timer.

Format

```
Set Idletime <number-of-seconds>
```

Parameters

number-of-seconds

Specifies the number of seconds NetEx waits before sending an idle message to the remote host. The default is 6. (Range is 1-999)

SET IPROUTE

Description

Create an IP routing table entry.

Format

```
Set IProute <GNA> <ipv4-address>
```

Parameters

GNA

The NCT defined GNA (NETADDR and SMGDREF) of a host adapter.

ipv4-address

Normal dotted decimal IP address format: xxx.xxx.xxx.xxx

Notes

Please note that all IP routing table updates made with this command are valid only as long as NetEx/IP is running. These changes will be lost should NetEx/IP be recycled. If the operator command 'LOAD NCT' is issued, the operator entered entry will only be replaced if there is a successful DNS lookup for that GNA.

SET MAXBI

Description

Specifies the maximum input buffer size that a user may specify on a CONNECT or OFFER call. This parameter sets a system wide maximum user buffer size. Its value and the size of the user buffer region determine possible fragmentation of the region and the maximum number of connections that can be supported.

Format

```
Set MAXBI <number-of-bytes>
```

Parameters

number-of-bytes

Specifies the maximum input buffer size (in bytes) that users can specify on a CONNECT or OFFER call. Size may be from 2048 to 65400 bytes, but must be greater than or equal to the default block-in and SEGSIZE values. The default is 65400.

SET MAXBO

Description

Specifies the maximum output buffer size that a user may specify on a CONNECT or OFFER call. This parameter sets a system wide maximum user buffer size. Its value and the size of the user buffer region determine possible fragmentation of the region and the maximum number of connections that can be supported.

Format

```
Set MAXBO <number-of-bytes>
```

Parameters

number-of-bytes

Specifies the maximum output buffer size (in bytes) that users can specify on a CONNECT or OFFER call. Size may be from 2048 to 65400 bytes, but must be greater than or equal to the default block-out and SEGSIZE values. The default is 65400.

SET MAXKBS

Description

Sets the maximum rate at which NetEx/IP will deliver data to the network for each network connection. If zero is specified, data will be delivered with no internal throttling. This value is only used for connections to hosts that do not have a ‘rate’ value specified in the PAM.

Format

```
Set MAXKBS <number-of-kilobits> [i]
```

Parameters

number-of-kilobits

Specifies the maximum rate in Kbits per second. For example, a value of 50 means 50 Kbs; a value of 50000 means 50 Mbs (50,000 Kbs). The default is 0, indicating no limit. (Range is 0-2147483647)

i (immediate)

Specifies the new setting should be applied to established connections. If this indicator is not specified, only connections established after maxkbs is set will use the new setting.

SET MBUFIN

Description

Sets the maximum number of buffers used for incoming data, unless overridden by rate, delay, and/or segsize considerations.

Format

```
Set MBUFIN <number-of-buffers>
```

Parameters

number-of-buffers

Specifies the number of buffers. The default is 8. (Range is 0-2147483647)

SET MBUFOUT

Description

Sets the maximum number of buffers used for outgoing data, unless overridden by rate, delay, and/or segsize considerations.

Format

```
Set MBUFOUT <number-of-buffers>
```

Parameters

number-of-buffers

Specifies the number of buffers. The default is 8. (Range is 0-2147483647)

SET MSGLVL

Description

Controls the severity of messages printed on the operator's console. The operator messages have seven levels. . All messages with the specified level of severity or greater are displayed.

Format

```
Set MSGLvl <level>
```

Parameters

level

Specifies a keyword to indicate the level of messages to be displayed on the operator's console. Specify one of the following:

immediate	Messages that require immediate action by the operator. Example: NetEx termination.
important	Messages that are of great interest to the operator and may require operator action. Examples: notification of all set drain, start, and clear commands; remote operator messages.
interesting	Messages regarding events that are of interest in monitored environments. This is the default message level.
moderate	Messages regarding more significant events that are of interest in monitored environments noting specific events under normal circumstances
monitor	Messages regarding events that are of interest in heavily monitored environments.
debug	Messages that are intended for diagnosing a particular problem. These messages are generally only used by support attempting to diagnose a specific NetEx problem
blither	Messages that are intended for diagnostic or debugging purposes. These messages are generally only of interest when a system programmer is attempting to diagnose a NetEx problem.

SET MSGSYSLOG

Description

Set where NetEx msgs will be output

Format

```
Set MSGSyslog <value>
```

Parameters

value

0, 1. A value of 0 means minimal output to the NetEx log. A value of 1 means output to the NetEx log.

The default is 1.

SET MULTIHOST

Description

Specifies whether multihost is enabled or disabled. This parameter is important when using TNP.

Format

```
Set MULTIhost {on | off}
```

Parameters

on | off

Specifies whether multihost is enabled or disabled. Specify ON to enable and OFF to disable.

The default is OFF.

SET NTXOPER

Description

Specifies whether the remote operator service is enabled or disabled.

Format

```
Set NTXOPer {on | off}
```

Parameters

on | off

> Specifies whether the remote operator service is enabled or disabled. Specify ON to enable the remote operator service; specify OFF to disable the remote operator service. The default is ON.

SET POLLSEL

Description

Specifies whether to poll with no timeout or not.

Format

```
Set POLLSel {on | off}
```

Parameters

on | off

> Specifies whether the polling on the sockets will include a timeout. If ON it will not include the timeout in when deciding when to poll. If OFF, it will use the calculated timeout. Set it to ON when sending smaller segments. The default is ON.

SET PREFPROT

Description

Sets the default preferred protocol type to use when connecting to hosts that support multiple NetEx/IP protocol types.

Format

```
Set PREFPROT {2 | 4}
```

Parameters

2 | 4

Specifies the default preferred protocol type. Specify 2 to use type-2 as the default protocol; specify 4 to use type-4 as the default protocol. The default is 2.

SET RCVDATAQHB

Description

This is the high threshold value (in bytes) for the size of the receiving NetEx DataQue for each network connection. If the size of the Data Queue exceeds this value, subsequent blocks received on the network that are greater than the highest received LRN are NAK'ed and discarded. Subsequent blocks will continue to be discarded until the size of the DataQue is reduced to the value specified by 'rcvdataqlb'. (Prot 4 only).

Format

```
Set RCVDATAQHB <number-of-bytes>
```

Parameters

number-of-bytes

Specifies the number of bytes. The default is 20000000. (Range is 0-2147483647)

SET RCVDATAQLB

Description

This is the low threshold value (in bytes) for the size of the receiving NetEx DataQue for each network connection. See the ‘rcvdataqhb’ parameter for the description of how this value is used. (Prot 4 only)

Format

```
Set RCVDATAQLB <number-of-bytes>
```

Parameters

number-of-bytes

Specifies the number of bytes. The default is 10000000. (Range is 0-2147483647)

SET RCVDATAQHS

Description

This is the high threshold value (in segments) for the size of the receiving NetEx DataQue for each network connection. If the size of the Data Queue exceeds this value, subsequent blocks received on the network that are greater than the highest received LRN are NAK’ed and discarded. Subsequent blocks will continue to be discarded until the size of the DataQue is reduced to the value specified by ‘rcvdataqls’. (Prot 4 only).

Format

```
Set RCVDATAQHS <number-of-segments
```

Parameters

number-of-segments

Specifies the number of segments. The default is 15000. (Range is 0-2147483647)

SET RCVDATAQLS

Description

This is the low threshold value (in segments) for the size of the receiving NetEx DataQue for each network connection. See the ‘rcvdataqhs’ parameter for the description of how this value is used. (Prot 4 only)

Format

```
Set RCVDATAQLS <number-of-segments>
```

Parameters

number-of-segments

Specifies the number of segments. The default is 8000. (Range is 0-2147483647)

SET REXMWBLKS

Description

If userexmitq is being used, this is the number of blocks to use to calculate the time to wait before rexmitting a block when it is NAKed.

Format

```
Set REXMWBlks <number-of-blocks>
```

Parameters

number-of-blocks

Specifies the number of segments. This number * segment size / rate will give the time to wait.

The default is 2. (Range is 0-2147483647)

SET READTIME

Description

Specifies the number of seconds NetEx transport retains user data waiting for the receiver to issue a READ request. When this timer expires a disconnect will be flagged and sent to the remote process connected. The local process will be sent a disconnect message for READTIME seconds, if there is not already one there. When a disconnect times out, the transport connection will be cleared out and the Tref will become invalid for future user requests.

Format

```
Set READtime <number-of-seconds>
```

Parameter

number-of-segments

Specifies the number of seconds (1-99999) NetEx transport retains user data waiting for a READ to be issued from the receiver. The default is 30.

SET ROPCLASS

Description

Specifies the class of operator commands that the remote operators will be allowed to issue.

Format

```
Set ROPClass {a|g}
```

Parameters

a | g

Specifies a value to indicate the class of remote operator commands to be available to remote hosts as follows:

A indicates that all commands are allowed.

G indicates that only display commands are allowed. This is the default.

SET SESMAX

Description

Controls the number of session connections or OFFERs permitted at one time. If the current number of sessions is greater than the new value specified, the command will not affect sessions in progress but will deny any new requests until sessions are disconnected. If the current number of sessions is less than the new value, then there will be no immediate effect.

Format

```
Set SESMax <number-of-sessions>
```

Parameters

number-of-sessions

Specifies the number of connections and OFFERs to allow at one time (from 2 to the 255 value).

The default is 32.

SET USERCVGAPQ

Description

This specifies whether or not to use the receive gap queue. Currently it should be always set to 0.

Format

```
Set USERCVgapq { 0 | 1 }
```

Parameters

0 | 1

0 means do not use the receive gap queue, 1 means use it. The default is 0.

SET USEREXMITQ

Description

This specifies whether or not to use the rexmit queue. NAKed blocks will queue for a certain amount of time determined by segsize and rexmwblks before being rexmitted. This can allow an ACK to come in during that time and avoid a retransmission. Useful when block are being delivered out of order on the network.

Format

```
Set USEREXmitq { 0 | 1 }
```

Parameters

0 | 1

> 0 means do not use the rexmit queue, 1 means use it. The default is 1.

SET WDOGINT

Description

Specifies the number of seconds that elapse between NetEx's checking for timed-out conditions in the NRB requests. If a READ has a timeout value specified as 10 seconds, and the WDOGINT is also 10 seconds, the READ will actually timeout in the range 10-20 seconds.

Format

```
Set WDogint number-of-seconds
```

Parameter

number-of-seconds

> Specifies the number of seconds that NetEx uses as a base unit for checking time out values. (Range is 1-999)
>
> The default is 2.

SET XDBG

Description

This is a special debug option to see details of throttling.

Format

```
Set XDbg <number>
```

Parameters

number

Setting the value > 0 will turn on xdbg tracing, default is 0..

START ADAPTER

Description

Start a DRAINED adapter.

Format

```
STart Adapter <GNA>
```

Parameters

GNA

The four HEX character network address (GNA or DREF) of the adapter to be started.

START HOST

Description

Starts a remote host which had been drained.

Format

```
START HOST <hostname>
```

Parameters

hostname

Specifies the name of the remote host to be started.

START NETEX

Description

Restarts NetEx after it has been drained using the DRAIN NETEX command.

Format

```
START NETEX
```

START PATH

Description

Starts a path which had been drained.

Format

```
START PATH <from-gna> <to-gna>
```

Parameters

from-gna

Specifies the local GNA of the path to be started.

to-gna

Specifies the remote GNA of the path to be started.

SWLOG

Description

Save and close the current ‘ntxlog’ file. NetEX will start using a new ‘ntxlog’ file

Format

```
SWLOG
```

Parameters

None.

Appendix A: NRB Error Codes

When a NetEx request is issued, the results of the request are returned in one or both of two NRB fields, NRBSTAT and NRBIND. These fields are located at the beginning of the NRB to make their subsequent examination by high level language programs a simpler matter.

NRBSTAT is designed to indicate if an operation is in progress and whether it completed successfully. NRBIND is designed to indicate the type of information that arrived as the result of a read type command (OFFER or READ).

When the operation is accepted by the NetEx user interface, the value of NRBSTAT is set to a –1. Thus, the sign of this word is an "operation in progress" flag for all implementations.

If an operation completed successfully, NRBSTAT is returned as all zeroes. If a read–type command was issued, then an "indication" is set in NRBIND. The termination of a session is always indicated by a disconnect indication in NRBIND regardless of the request type.

If the operation did not complete successfully, then NRBSTAT contains a standard error code. NRBSTAT is represented as four decimal digits. The thousands digit denotes the origin of the error; the low order three digits identify the error type. The codes for error origin are as follows.

Code	**Description**
0xxx	NetEx general. Errors detected by the user interface that prohibit proper process of the command.
1xxx	Driver level errors.
2xxx	Transport level errors.
3xxx	Session level errors.
4xxx	Network level errors.
5xxx-8xxx	Reserved for future NetEx functions
90xx	Reserved for errors returned by user exits on the local host
91xx	Reserved for errors returned by user exits on the remote host during the connection process.
9200-32767	Reserved for future NetEx functions.

Table 2. Origin of NRB Error Codes

0xxx and 90xx errors can be returned to any user program that accesses any level of NetEx services. Normally, an application that accesses services at say, transport level receives only those errors (2xxx) related to transport services. However, the principle within NetEx is that if a level elects to abort the user's request based on an error returned by a lower level of software, then the error code should be "rippled up" to the user rather than summarized at the higher level. For example, driver might report a "power off" or "not operation" status to transport if there is an adapter failure. If transport decided that this error type should cause loss of communications, then the 1xxx error is returned to the user along with a Disconnect Indication in NRBIND when the next user read command was issued.

Following that, the second digit places the errors in categories:

Digit	**Category**
x0xx	NetEx general or inconsistent NRB formats
x1xx	Specification errors in parameters passed to a particular protocol level
x2xx	Hardware errors
x3xx	Request out of sequence and read timeouts
x4xx	NetEx Initiated disconnect errors
x5xx	Errors during connection

Table 3. Origin of NRB Error Codes, Part 2

Note the following when using these codes:

The error codes at each level are as common as possible. Thus, a 2103 error in transport would have substantially the same meaning as a 3103 error in session, and a 1361 error would not be defined at (for example) the Driver level if a 3361 error meant something entirely different at the Session level.

Some errors cause the loss of the connection or result in a connection not being established. Any status code that implies that the connection is no longer useful has a 6 (Disconnect Indication) returned in NRBSTAT. Any attempts to issue further requests to that connection have a x100 (no Nref) error returned to it.

All errors that result in the loss of the connection and a Disconnect Indication in NRBIND are indicated by an asterisk (*) following the error code number.

Note: A 0000 in field NRBSTAT means successful completion of the NetEx request. A -1 means that the request is still in progress.

The following sections describe the errors in numerical order starting with general NetEx error, followed by driver, transport, and session level errors.

General Errors

The following errors are general NetEx errors.

0000	Successful completion
0001	A read type operation completed normally within NetEx, but the Pdata buffer provided by the user was not large enough to hold the data. NRBLEN and NRBUBIT reflect the amount of data the other party intended to send. However, the amount of data moved to the user's program was only the amount of addressable units specified in NRBBUFL. NRBIND specifies the type of data sent to the user. Requests affected: OFFER, READ. The status of the connection is not affected.
0002	NRBBUFL and NRBBUFA do not specify a block of storage that fits entirely within the user's addressable memory. The operation is suppressed. All requests are affected. The status of the connection is unaffected.
0003	On a write type operation, the unused bit count(NRBUBIT) specifies a larger number of bits than are in the machine's word (addressable unit size). The operation is suppressed; the status of the connection is not affected.
0004	The request code(NRBREQ) is not valid. The operation is ignored, and the status of the connection specified by NRBNREF is not affected

0005	The buffer size specified (in NRBBUFL for read and NRBLEN for write) exceeds an implementation defined NetEx maximum. The operation is suppressed. The status of the connection is not affected.
0011	A read-type operation completed normally within NetEx, but the Odata buffer provided by the user was not large enough to hold the data. NRBPROTL reflects the amount of data the other party intended to send; however, the amount of data moved to the user's program was only the amount of addressable units originally specified in NRBPROTL. NRBIND specifies the type of data sent to the user. Requests affected: OFFER, READ. The status of the connection is not affected
0012	NRBPROTL and NRBPROTA do not specify a block of storage that fits entirely within the user's addressable memory. The operation is suppressed. All requests are affected. The status of the connection is unaffected.
0021	A read-type operation completed normally within NetEx, but BOTH the Odata and Pdata buffers were too small to hold the incoming data. NRBLEN/NRBUBIT and NRBPROTL reflect the amount of data the other party intended to send; however, the amount of data moved to the user's program was only the amount of addressable units originally specified in NRBLEN and NRBPROTL. NRBIND specifies the type of data sent to the user. Request affected: OFFER, READ. The status of the connection is not affected.
0100*	The user interface detected that the NRBNREF in the NRB does not refer to a connection currently in use by the application program. Probable cause is a bad CONNECT, OFFER, or ASSIGN or failure to handle an incoming Disconnect.
0310	The user has attempted to re-use an NRB before a previous request issued with that NRB has completed. The request will be rejected. When the original request issued with that NRB completes, then the NRB will be once more updated with the status of that request.
0500*	NetEx is not currently running on the local computer. Intervention by the local computer operator will be required to start NetEx. This code is issued by the NetEx user interface when it is determined that NetEx is unavailable.
0503*	An OFFER, CONNECT, or ASSIGN request has resulted in the number of connections outstanding for the caller exceeding an implementation defined maximum. The new connection request is rejected.
0504*	The user program is not authorized to use the user interface facilities needed to communicate with NetEx. No use of NetEx is possible until the user gains the appropriate authorization
0505*	NetEx is currently being drained by the computer operator in preparation for a NetEx shutdown. No new OFFER, CONNECT, or ASSIGN requests will be accepted. The request is rejected. The status of already existing connections is not affected.
0511*	A CONNECT or ASSIGN request would exceed the total number of driver service level connections to NetEx. The new connection request is rejected.
0512*	The NetEx program is aborting execution due either to internal NetEx software problems or cancellation by the computer operator. No further traffic with NetEx will be possible. This error will be issued to complete a request that was issued when NetEx was running normally.

Table 4. General NRB Error Codes

Host Specific Errors

0913	The number of concurrent NetEx requests has exceeded the number of available NetEx NRBs. The request can be retried at a later time.

Table 5. Host Specific NRB Error Codes

License Specific Errors

0600	License does not support IP
0601	License does not support HyperChannel (NESiGate)
0602	License does not support protocol 4
0610	Can't read license
0611	License is invalid
0612	License has expired
0613	License does not support TNP

Table 6. License Specific NRB Error Codes

Driver Service Errors

1005	The length of data on a DWRITE is greater than a host-specified maximum. The request is rejected.
1006	The length of data received on a DREAD is greater than a host-specified maximum. The request is rejected.
1100*	The Dref specified by the NRBNREF is not in use or is not owned by this application program. The request is rejected. The status of the other connections owned by this application remains unchanged.
1101	The DATAMODE field of a NetEx format network message is not valid for this particular host; or, the message is routed through the driver (intra-host) and Assembly/Disassembly cannot be performed.
1102	The specified value of the Associated Data bit in the hardware message area does not match the presence or absence of associated data as specified in NRBLEN. DWRITW is affected. The driver assignment remains in effect. Both NetEx format and arbitrary format network messages are affected.
1103	The specified length of the message proper does not fit, it is not between 8 to 64 bytes inclusive. Only writes (DWRITE) may obtain this response. The driver assignment remains in effect. Both NetEx format and arbitrary format network messages are affected.
1200	"Power off", "not operational", or a similar indication of local adapter unavailability was discovered when physical I/O was issued. The status of the assignment is not affected, but it is unlikely that driver communications can continue without operator intervention.
1201	The network adapter has reported an error in processing the DREAD or DWRITE request. The adapter model dependent detailed status may be obtained by issuing a DSTATUS function.

1300	A DREAD or DCONNECT request timed-out before any data was received on the network. The time value used for the timeout was in NRBTIME. No data was received. The status of the driver connection is not affected.
1304	The number of DWRITE requests outstanding against a single connection exceeds an implementation defined maximum. The DWRITE request is rejected. The status of the connection and the previous DWRITE requests remains unchanged.
1305	The number of DREAD requests outstanding against a single connection exceeds an implementation defined maximum. The DREAD request is rejected. The status of the connection and the previous DREAD requests remains unchanged.
1306	A DREAD or a DWRITE was detected when the connection was in disconnect mode.
1310	The device service was forced to discard the Associated Data segment of a message because no DREAD was issued within a sufficient time of the arrival of the network message. The message proper is returned to the user. Also, a DWRITE will receive this error if an intra-host DWRITE cannot be matched with an outstanding DREAD by another driver user. In that case, the message proper will be queued and associated data discarded.
1311	Message proper's have been lost because of excess demand for the Driver service's resources. One or more messages that arrived before the current message were totally discarded by the driver service. This will be provided as a DREAD error or an intra-host DWRITE.
1312	A request for a privileged service such as diagnostic mode has been issued to the driver. The request is rejected as the user does not have sufficient implementation dependent privileges. This error applies to both DREAD requests and intra-host writes (DWRITE).
1501*	A specific Dref requested by the DCONNECT is already in use. If a nonspecific request was made, all driver paths are in use.
1503*	The number of user driver attaches permitted by NetEx has been exceeded. Driver service cannot be offered at this time. The DCONNECT is rejected.
1504*	Driver service is not directly available to applications programs. This service can only be made available by the installation systems programmer.
1505*	NetEx is currently being "drained" by the computer operator. No new driver service (DCONNECT) requests are being accepted.
1506*	The specific Dref (adapter address) requested by a DCONNECT does not exist on this local host.
1507*	The specific Dref (adapter address) exists on the local host, but the NetEx operator has drained that adapter so no new requests for driver service (DCONNECT requests) can be accepted on that adapter.
1509*	The specified value of NRBBLKO exceeds an installation or implementation defined maximum. The DCONNECT request is rejected.
1510*	The specified value of NRBBLKI exceeds an installation or implementation defined maximum. The DCONNECT request is rejected.
1601	Hardware error – lost interrupt.
1605	Driver initialization failed.

1606	NTX minor device in use.
1612	Cannot attach to shared memory.
1614	NRB not in shared memory.
1628	Driver is out of buffer space.

Table 7. Driver Service NRB Error Codes

Transport Service Errors

2005	During a WRITE operation, the length of the buffer as specified by NRBLEN exceeds the maximum buffer size found in NRBBLKO. The WRITE operation is rejected. The connection remains outstanding.
2008	During a segmented write, NRBLEN exceeds the segment size.
2100*	The Tref specified by NRBNREF is not in use or is not owned by this application program. The request is rejected. The status of the other connections owned by this application remains unchanged.
2101	The DATAMODE field in the NRB is not valid for the local host. The write operation (SWRITE, TWRITE, SCONNECT, TCONNECT, SCLOSE, TCLOSE) is suppressed. The connection (if previously established) remains in effect.
2103	The quantity of Odata provided exceeds an implementation defined maximum. The request is rejected.
2300	The timeout value associated with a TREAD request resulted in a request timing out before any data or other indication was received from the corresponding application.
2301	TCONNECT, TOFFER, or TCONFIRM has been issued for a connection that is already fully established. The request is rejected. The status of the connection remains unchanged. Some implementations may return a 2301 code for any “out of sequence” series of request to Transport.
2302	A connect indication was received by a preceding offer, and a request other than TCONFIRM or TDISCONNECT was issued. The request is rejected. NetEx will continue to wait for the confirm or disconnect request.
2303	A TCONNECT request was previously issued, then a request other than a TREAD to read the Confirm of Disconnect indication was issued. The request is rejected. NetEx will continue to wait for the TREAD request.
2304	The number of TWRITE requests outstanding against a single connection exceeds an implementation defined maximum (usually one). The TWRITE request is rejected. The status of the connection and the previous TWRITE requests remains unchanged.
2305	The number of TREAD requests outstanding against a single connection exceeds an implementation defined maximum (usually one). The TREAD request is rejected. The status of the connection and the previous TREAD requests remains unchanged.
2306	A TWRITE request has been issued to a transport connection that is in the process of servicing a remote caller or NetEx initiated Disconnect. A Disconnect Indication is pending from NetEx.
2307	A TREAD request has been issued to a transport connection that is in the process of servicing a remote caller or NetEx initiated Disconnect. A Disconnect Indication is pending from NetEx.
2308	A write type request (TWRITE or another TCLOSE) has been issued against a connection that has accepted a previous TCLOSE.

2400*	No response has been received from the remote NetEx for a period of elapsed time (DEADTO) specified by the installation systems programmer. The connection is terminated. A Disconnect Indication will be found in the NRBIND.
2402*	The remote application has failed to issue a TREAD request for a period of elapsed time (READTO) specified by the installation systems programmer on the remote host. The connection is terminated. A Disconnect Indication will be found in NRBIND.
2403	The remote application exited without issuing an explicit Disconnect back to the local application. The connection is terminated. A Disconnect Indication will be found in NRBIND.
2500*	A connect message was repeatedly sent to the remote host in response to a previous TCONNECT request, but no response was received for a period of elapsed time (CONTO) specified by the installation systems programmer.
2503*	The number of transport connections permitted by NetEx has been exceeded. Transport service cannot be offered at this time. The TCONNECT or TOFFER is rejected.
2504*	Transport service is not directly available to applications programs. This service can only be made available by the installation systems programmer.
2505*	NetEx is currently being “drained” by the computer operator. No new request for Transport services (TCONNECT of TOFFER requests) are being accepted.
2506*	The Physical Address Map passed to Transport for a connection is not valid. If this message was returned from a SCONNECT request, the Network Configuration list was incorrectly generated.
2509	The specified value of NRBBLKO exceeds an installation or implementation defined maximum. The connection request is rejected.
2510*	The specified value of NRBBLKI exceeds an installation or implementation defined maximum. The connection request is rejected.
2511*	The specified Class of Service is not implemented.

Table 8. Transport Service NRB Error Codes

Session Service Errors

3005	During a WRITE operation, the length of the buffer as specified by NRBLEN exceeds the maximum buffer size found in NRBBLKO. The WRITE operation is rejected. The connection remains outstanding.
3006	The length of PDATA sent on a CONNECT, CONFIRM, or DISCONNECT is greater than the maximum allowed. The request is rejected.
3008	During a WRITE, NRBLEN exceeds segment size.
3100*	The Sref specified by NRBNREF is not in use or is not owned by this applications program. The request is rejected. The status of other connections owned by this application remains unchanged.
3101	On an SWRITE request for intra-host communications, a DATAMODE was specified that is not supported for internal communications.
3103	The quantity of Odata provided exceeds an implementation defined maximum. The request is rejected.
3300	An SREAD or SOFFER request timed-out before a response was received on the network. If the timed request is an SREAD, the status of the connection was not affected. If an SOFFER timed out, then the connection will not have taken place.
3301	SCONNECT, SOFFER, or SCONFIRM has been issued for a connection that is already fully established. The request is rejected. The status of the connections remains unchanged.
3302	A connect indication was received by a preceding SOFFER, and a request other than SCONFIRM or SDISCONNECT was issued. The request is rejected. NetEx will continue to wait for the confirm or disconnect request.
3303	A SCONNECT request was previously issued, then a request other than an SREAD or SDISCONNECT was issued. The request is rejected. NetEx will continue to wait for the SREAD or SDISCONNECT request.
3304	The number of SWRITE requests outstanding against a single connection exceeds an implementation defined maximum (usually one). The SWRITE request is rejected. The status of the connection and the previous SWRITE requests remains unchanged.
3305	The number of SREAD requests outstanding against a single connection exceeds an implementation defined maximum (usually one). The SREAD request is rejected. The status of the connection and the previous SREAD requests remains unchanged.
3306	A SWRITE has been issued to a session connection that is in the process of servicing a remote caller or NetEx initiated Disconnect. A Disconnect Indication is pending from NetEx.
3307	A SREAD request has been issued to a session connection that is in the process of servicing a remote caller or NetEx initiated Disconnect. A Disconnect Indication is pending from NetEx.
3308	A write type request (SWRITE or another SCLOSE) has been issued against a connection that has accepted a previous SCLOSE.

3402*	The remote application has failed to issue an SREAD request for a period of elapsed time (READTO) specified by the installation systems programmer on the remote host. The connection is terminated. A Disconnect Indication will be found in NRBIND.
3403*	The remote application exited without issuing an explicit Disconnect back to the local application. The connection is terminated. A Disconnect Indication will be found in NRBIND
3404	Explanation: The write didn't complete within WRITETO seconds (may have been caused by the session manager).
3422	HALT SREF was issued by operator.
3500*	A connect message was repeatedly sent to the remote host in response to a previous TCONNECT request, but no response was received for a period of elapsed time (CONTO) specified by the installation systems programmer. Probable cause is the absence of the NetEx software on the remote host. The SCONNECT terminates with a Disconnect Indication in NRBIND.
3501*	The PNAME specified is not OFFERed on the HOST specified during the SCONNECT. The SCONNECT terminates with a Disconnect Indication in NRBIND.
3502*	The PNAME specified was not OFFERed on the HOST specified during the SCONNECT. However, a session that was previously established by OFFERing the requested PNAME is now in progress on the remote machine. If the remote application elects to re-OFFER the connection in the future the service might be available at that time. (In other words, the remote application is “busy.”)
3503*	The number of user session connections permitted by NetEx has been exceeded. Session service cannot be offered at this time. The SCONNECT or SOFFER is rejected.
3504*	Session service is not directly available to applications programs. This service can only be made available by the installation systems programmer.
3505*	NetEx is currently being “drained” by the computer operator. No new requests for Session services (SCONNECT and SOFFER) are being accepted.
3506*	The HOST specified in an SCONNECT request does not exist anywhere on the network generated by the installation systems programmer. The SDISCONNECT terminates with a Disconnect Indication in NRBIND.
3507*	The HOST specified exists on the installation generated network configuration, but the local computer operator has specified that no session level connections take place with that particular host. The SCONNECT terminates with a Disconnect Indication in NRBIND.
3508*	The HOST specified exists on the installation generated network configuration, but no communications path exists between the local host and the specified remote host. The SCONNECT terminates with a Disconnect Indication in NRBIND.
3509*	The specified value of NRBBLKO exceeds an installation or implementation defined maximum. The connection request is rejected.
3510*	The specified value of NRBBLKI exceeds an installation or implementation defined maximum. The connection request is rejected.
3511*	The Class of Service request is not currently implemented.
3522*	NetEx was drained while this outstanding OFFER was not complete.

3523*	NetEx was DRAINed when a connect was received. This error is returned by the Session Manager to the connector.
3550*	The local host specified on an SOFFER or SCONNECT does not exist in the NetEx Administrator’s NCT. The request is rejected at the Administrator.
3552*	The local host specified on an SOFFER or SCONNECT request in not in the NetEx Administrator’s domain. The request is rejected.
3553*	The Physical Address Map (PAM) sent along with an OFFER or CONNECT request to the Administrator does not match any PAM that the Administrator can generate. The request is rejected.

Table 9. Session Service NRB Error Codes

Network Service Errors

4100*	The Nref specified by NRBNREF is not in use or is not owned by this application program. The request is rejected. The status of other connections owned by this application remains unchanged.
4101	In a Network connection that is intra-host (causing no network adapter traffic) a DATAMODE was requested on the NWRITE that is not supported for intra-host communications. The block will be sent to the destination process using bit stream (DATAMODE 0) transmission.
4104	Checksum on an incoming driver level message is not correct. The message and data received will be returned to the DREAD caller along with the error code but the data should, of course, be considered suspect. The status of the driver assignment is not affected.
4105	The length of the Pdata was less than or substantially different from the specified length in the message proper. This comparison is performed after adjustment for incoming A/D modes. Sufficient slop in this comparison will be done to accommodate those machines that must send information in multiples of the word size.
4300	The timeout value associated with an NREAD request resulted in a request timing out before any data or other indication was received from the corresponding application.
4301	NCONNECT or NOFFER has been issued for a connection that is already fully established. The request is rejected. The status of the connection remains unchanged. Some implementations may return a 4301 code for any "out of sequence" series of requests to Network Service.
4304	The number of NWRITE requests outstanding against a single connection exceeds an implementation defined maximum (usually one). The NWRITE request is rejected. The status of the connection and the previous NWRITE requests remains unchanged.
4305	The number of NREAD requests outstanding against a single connection exceeds an implementation defined maximum (usually one). The NREAD request is rejected. The status of the connection and the previous NREAD requests remains unchanged.
4306	A NWRITE has been issued to a transport connection that is in the process of servicing a NetEx initiated Disconnect. A Disconnect Indication is pending from NetEx.
4307	A NREAD request has been issued to a transport connection that is in the process of servicing a NetEx initiated Disconnect. A Disconnect Indication is pending from NetEx.
4403*	When processing an NWRITE request, Network Service found that a network Virtual Circuit between two Network applications no longer exists. The Network connection is terminated.
4501*	A specific Nref requested by the NCONNECT or NOFFER is already in use.
4503*	The number of user Network connections permitted by NetEx has been exceeded. Network service cannot be offered at this time. The NCONNECT or NOFFER is rejected.
4504*	Network service is not directly available to applications programs. This service can only be made available by the installation systems programmer.
4505*	NetEx is currently being "drained" by the computer operator. No new requests for Network services (NCONNECT or NOFFER requests) are being accepted.

4506*	The Physical Address Map passed to Network for a connection is not valid. If this message was returned from an SCONNECT request the Network Configuration list was incorrectly generated.
4509*	The specified value of NRBBLKO exceeds an installation or implementation defined maximum. The connection request is rejected.
4510*	The specified value of NRBBLKI exceeds an installation or implementation defined maximum. The connection request is rejected
4511*	The specified Class of Service is not implemented.
4512*	During an attempt to establish a virtual circuit on HYPERbus or other equipment, a component of the network physically did not respond. The circuit cannot be established.
4513*	During an attempt to establish a virtual circuit on HYPERbus or other equipment, a component of the network could not honor the request because all its circuit facilities were "busy." The circuit cannot be established at the current time.
4514*	During an attempt to establish a virtual circuit on HYPERbus or other equipment, a component of the network could not honor the request because of an equipment failure.
9001	User exit rejected request (Completion). The request has been failed by the user interface because NetEx has completed processing.

Table 10. Network Service NRB Error Codes

Appendix B: Messages

This section contains a description of the messages issued by NetEx/IP. These messages are displayed in the ‘ntxlog’.

Each message is prefixed with a date and timestamp of the following format:

```
    Sun Jan 27 01:03:52 2008
```

Messages:

```
ISOFFR nref n uname    : offering pname at hname, t secs
```

Description: An OFFER has been issued by the NetEx/IP application that is running under the user name ‘uname’.

n	identifies the internal NetEx/IP session reference number
uname	identifies the user name under which the NetEx/IP application is running.
pname	identifies the NetEx/IP OFFER name of the application.
hname	identifies the name of the host on which the OFFER occurred (local host)
t	identifies the OFFER timeout value specified by the application.

User Response:

None.

```
ISCONN nref n uname    : connecting to pname at hname,
path from aa to bb
```

Description: A CONNECT has been issued by the NetEx/IP application that is running under the user name ‘uname’.

n	identifies the internal NetEx/IP session reference number
uname	identifies the user name under which the NetEx/IP application is running.
pname	identifies the NetEx/IP name of the remote application (remote OFFER name).
hname	identifies the name of the host being connected to (remote host).
aa	identifies the local unit portion of the network path address
bb	identifies the remote unit portion of the network path address

User Response:

None.

```
ISCONF nref n uname    : pname confirming to nref m at hname
```

Description: An OFFER has been completed (connected to) by a remote NetEx/IP application, and the subsequent CONFIRM has been issued.

n	identifies the internal NetEx/IP session reference number of the local application.
uname	identifies the user name under which the NetEx/IP application is running.
pname	identifies the NetEx/IP OFFER name of the application.
m	identifies the internal NetEx/IP session reference number of the remote application
hname	identifies the name of the host on which the OFFER occurred (local host)

User Response:

None.

```
ISCLOS nref n uname    : closing
```

Description: A CLOSE has been issued by the NetEx/IP application that is running under the user name ‘uname’.

n	identifies the internal NetEx/IP session reference number of the local application.
uname	identifies the user name under which the NetEx/IP application is running.

User Response:

None.

```
ISDISC nref n uname    : disconnecting
```

Description: A DISCONNECT has been issued by the NetEx/IP application that is running under the user name ‘uname’.

n	identifies the internal NetEx/IP session reference number of the local application.
uname	identifies the user name under which the NetEx/IP application is running.

User Response:

None.

```
FINISHSESSION nref n : nrbreq = rr, nrbstat = ssss, nrbind = i
```

Description: A NetEx connection is terminated, due to the reason indicated in the message.

n	identifies the internal NetEx/IP session reference number of the local message.
rr	identifies the NetEx/IP request type.
ssss	identifies the reason for the session failure. Refer to “Appendix A: NRB Error Codes” on page 125 for a description of the possible status codes.
i	identifies the status of the connection
	6 session is disconnected

User Response:

None.

```
NEXTPAM nref n : new path from aa to bb
```

Description: An APR (alternate path retry) operation has occurred for the indicated session connection.

n	identifies the internal NetEx/IP session reference number
aa	identifies the local unit portion of the new network path address
bb	identifies the remote unit portion of the new network path address

User Response:

None.

```
NETEX: network protocol error n
```

Description: A netex network protocol error has been detected. The specific error is identified by “n”

1	Attempt to reroute non-offering side
2	Attempt to confirm to offerer
3	Actual data shorter than specified in protocol
4	D-read failed
5	Not a NetEx message
6	Checksum error
9	Awaiting the second part of a two-part message after the first part was already received. We received a new one-part message instead.
10	Received the second part of a two-part message prior to receiving the first part.
11	Awaiting the second part of a two-part message after the first part was already received. We received the first part of a different two-part message instead.
12	Two-part message error—no P-data or O-data
13	Invalid network unique id
14	Bad status creating TUB NDB

User Response:

None. NetEx reports the invalid message and continues. This situation can sometimes occur when messages are dropped on busy networks.

```
NETEX: transport protocol error n
```

Description: A netex transport protocol error has been detected. The specific error is identified by "n"

1	-- base field too small
2	-- protocol level is not greater than or equal to 2
3	-- invalid message type
4	-- connect attempt to unlicensed PROT_4
5	-- tdb->tdblrn > tub->tubrplrn
6	-- 'disc' sub-field in 'x' msg type, 'type'
7	-- invalid sub-field type 'x', 'type'
8	-- sub-field 'x' too small = 'y', 'type', 'size'
9	-- sub-field 'x' length 'y' overlaps total transport length 'z', 'type', 'size', 'length'
10	-- New path has smaller segsize

User Response:

None. NetEx reports the invalid message and continues.

```
Message received with no read active
```

Description: A NetEx message was received on the network, but there was no NetEx 'read' request active at the time. Message is dropped.

User Response:

None. NetEx reports the condition and continues.

Repeated occurrences of this message could indicate the NetEx 'DREADQUEUE' parameter might need to be increased.

```
License initialization has failed, rc= cccc.
```

Explanation: The NetEx license initialization has failed. "**cccc**" specifies the reason code for the failure:

9001 : expiring

9002 : expired

9003: expired – product non-functional

9004 : license open error

9005 : invalid key

9007 : no fingerprint

Operator Response: Notify the person responsible for the NetEx installation.

Programmer Response: Verify that the software key is correctly installed. If the key is incorrect, contact Network Executive Software, Inc. to obtain the correct license key.

```
License verification has failed, rc= cccc.
```

Explanation: Specifies the reason code for the failure:

9001 : expiring

9002 : expired

9003: expired – product non-functional

9004: key file not found

9005 : invalid key

9007 : no fingerprint

Operator Response: Notify the person responsible for the NetEx installation.

Programmer Response: Verify that the software key is correctly installed. If the key is incorrect, contact Network Executive Software, Inc. to obtain the correct license key.

Appendix C: H300e Installation

Prerequisites

The following are hardware and software prerequisites for installing the H300e product.

- Unisys Dorado running OS2200.
- At least one other processor on the network running NetEx/IP software. This processor should be connected with another NetEx/IP (not required for intra-host test/evaluation).
- H300e can coexist in the same OS2200 partition as H300IPC. They will each need a separate license key, as they are different products.
- Customers must obtain a software KEY from NESi prior to running the H300e software. Keys are required to start H300e. Customers must contact NESi customer support with the customer site name, SYSINFO output, and the NetEx/IP product designator (e.g., H300e). NESi customer support will supply the necessary key once this information has been received. The customer needs to place this key into the NESikeys file as discussed later in this section.
- A low BDI number for the installation of a fixed gate subsystem. Refer to the current release of "Software Planning and Migration Overview" for your release of the OS2200 software.
- It is strongly recommended that a unique userid be created to install and run H300e. The USERID used to install H300e will become the owner of the fixed gate subsystem. The owner of the subsystem is required to stop the shared subsystem prior to starting netex. This insures a clean start of the shared subsystem, regardless of how H300e was terminate in the previous run.

All requirements for the equipment listed above must be met before proceeding with the installation.

Hardware Installation

Install and verify proper operation of the appropriate operating system.

Accessing the H300e Software Distribution

The H300e NetEx/IP software is available as a CFMT file. This file should be FTPed to your OS2200 system using the BIN, and QUOTE SITE CFMT options in ftp. Contact NESi Customer Support to request the download link.

Upgrading H300e

This is a new product. You must follow the "Software Installation" instructions.

Removing H300e

The product can be removed from your system, by using Solar. You would delete the product, the same as any other Unisys product. You will then need to delete any files used by H300e.

Software Installation

1. Tailor the install1 element in the release file that was FTPed to your OS2200 system regarding the package use the command:
    a. **Change the USERID and PASSWD to a valid user on your system. This userid should be used through the install process, including the solar install**.
    b. **This will also be the userid to start netex**.
    c. Review the file names and cycle limits.
        i. NTXD is the name of the netex that will be installed.
    d. Run the install1 job.
        i. This creates EXEC, LOG, CONFIG, and LOCK files.
2. In your COMAPI configuration file set MAX-UDP-BUFFERS to 1000.
    a. ADMIN KEYIN,APIA MAX-UDP-BUFFERS,1000
3. In your CPCOM configuration verify the IP Addresses you plan to use are assigned to the correct CPCOM. This includes a 127.0.0.n address used internally by H300E.
4. In your CONFIG file:
    a. In this sample, we use the netex host name (NTXD) in our file allocations. NetEx and BFX will need to be linked to the shared memory subsystem. If you have multiple copies of netex on the O2200 partition, you will need multiple EXEC files and multiple GATES files.
    b. Edit sym-ss-def in your CONFIG file.
        i. Change the FIXED BDI to your BDI number (0402)
        ii. Change the SEARCH to point to the GATES file (create by solar)
    c. Edit pd-sgs in your CONFIG
        i. Change the Product name to your netex host. (-SS is shared subsystem)
        ii. Change the SOURCE name to your CONFIG file.
        iii. Change the DEST to your netex host-ss (Should point at the GATES file).
        iv. Change the FILE name to your GATES file.
        v. Change the FIXED_Gate number to your extend-modeBDI-NUMBER (0200402)
    d. Do a solar local product install using the pd-sgs in your CONFIG file
        i. This will create the GATES file.
    e. Edit add-gates in the EXEC file.
        i. Change the use statement to point to the GATES file.
        ii. Change the use statement to point to the CONFIG file.
    f. @add the add-gates element

    @free your EXEC file

    g. Edit the link-netex element in the EXEC file. (START here for new software)
        i. Change the USE statement for the EXEC file
        ii. Change the USE statement for the GATES file.
        iii. Change the USE statement for the COMAPI file
        iv. Change the USE statement for the RELEASE file
        v. Submit the run
    h. Edit the link-eatgen element in the EXEC file. (OPTIONAL)
        i. Change the USE statement for the EXEC file
        ii. Change the USE statement for the GATES file.
        iii. Change the USE statement for the COMAPI file
        iv. Change the USE statement for the RELEASE file
        v. Submit the run

Post Installation Considerations

Configuring H300e

Once the software package installation has been successfully completed, NetEx/IP must be configured prior to execution. Sample elements can all be found in your EXEC file.

NETEXRUN	contains sample ECL to start the NetEx system.
CONSOLERUN	contains sample ECL to start the console interface
NTXDEFAULT	contains the NetEx initialization parameters.

The NETEXRUN, and CONSOLERUN should be copied to your SYS$LIB$*RUN$. These should be re-named to reflect the NetEx host name that will be started using this ECL. The RELEASE file should not be updated. Update the copy in your SYS$LIB$*RUN$ file.

NETEXRUN:

Change the userid to match the userid that was used to install the SUBSYSTEM on your system.
Change the password to be the valid password for the above userid.
Change the USE statement for the GATES file to reflect the name used in the install PD-SGS SHARED_SUBSYTEM.
Change the USE statement for the LOCK to match the LOCK file in the INSTALL1 run stream.
Change the LOG file ECL to reflect the destination of your NetEx Logfile
Change the BDI numbers to reflect the extend mode BDI number used at install. (3 places)
Change the CONFIG file to point to your netex configuration data.
Change the PAMFILE to point to the output from the Configuration Manager.

CONSOLERUN:

Change the EXEC file to point to your EXEC file.
Change the PRINT file to point to where you want the output from this run to be recorded to.
Change the DIAG file to point to where diagnostic information will be written to in case of an abend.

Update the configuration file if this is a new install. This configuration file is NOT compatible with H300IPC.

Copy in sample configuration parameters.

```
@copy,s  EXEC.NTXDEFAULT,CONFIG.NTXDEFAULT
@copy,s  EXEC.PRODCONF,CONFIG.PRODCONF
@copy,s  EXEC.PADSCONFIG,CONFIG.PADS$CONFIG
@copy,s  YOUR-NCT.,CONFIG.NCTNAM
@copy,a EXEC.CCTAB,CONFIG.CCTAB
```

Configuring the H300e NetEx/IP software consists of the following steps:

Step 1 Edit the ‘NESikeys’ file

Step 2 Configuring the default file

Step 3 Build an NCT

Step 4 Create NetEx/IP addressing information

Step 5 Build CCTABLE (optional)

Step 6 Start NetEx

Step 1. Edit the ‘NESikeys’ file

A single NESi License Key file must reside on each system where one or more NESi products containing license support will be installed. The following guidelines apply:

- The PRODCONF member in your CONFIG file points to the file containing the KEY element. The LICPATH keyword/value pair in the product configuration file specifies the full path name to this file. Update the LICAPTH in the PRODCONF to point to your CONFIG.KEY
- The default file name is *KEY*, this normally located in your CONFIG file.
- A leading ‘#’, ‘*’, ‘”’, ‘/’, or ‘!’ character, in a file record denotes a comment line.
- The systems programmer installing this software must edit this file to add a new encrypted Software Key each time such a key is obtained from NESi for H300e and/or other license-enabled NESi products. This should be done prior to installing the product, and must be done prior to any attempt to run the product successfully.
- To obtain a key from NESi, contact NESi support, supplying the fingerprint of the machine the software is to be installed on. This may be obtained by using the following ECL:

  @USE RELEASE.,NETEX*RELEASEFILE. (This was FTPED to your system.)

  @XQT RELEASE.SYSINFO

  (*This is the same information used in the H300IPC product*.)
- The file may contain multiple keys per product due to new product releases or a change to the platform’s fingerprint. If there are multiple keys the NESi product will use the first key found that matches the product name and system fingerprint starting at the top of the file. This makes it important to add new keys for an existing system to the top of the file. For example, if a new key is installed that provides a license date extension, or adds a new feature, adding this new key to the file before the old key ensures the new key will be used rather than the old key. To make the file easier to maintain over time, it is recommended that you precede each Key entry with a comment line that documents the product designator (e.g., H300e) and release level of the product that the key is associated with. It will then be easier to delete older Keys that are no longer valid for the product.

- To add the $KEY$ to your CONFIG,

  @ed,iq CONFIG.$KEY$   (This file must be in quarter word format)

  Insert a line and paste in the key supplied to you. It must start in column 1.

  Comments may be added.

- The following shows an example of what a *NESikeys* file might look like after adding a key to the file:

  ```
  # Network Executive Software, Inc. Software License Key file
  # Key for H300e R7.0.4:
  CGGZ-4AAA-AAAE-IAO5-O5OJ-SBHX-AUZ5-PL4D
  ```

Step 2. Configuring the default file

See Appendix D: NetEx Default File Configuration on page 155 to configure your CONFIG.NTXDEFAULT element.

In addition to Appendix D: NetEx Default File Configuration, Unisys users will need to configure the application interface to allow programs like ntxoper and BFX to communicate with a running version of NetEx. This involves 5 additional parameters in the ntxdefault file. Applications will request these values from NetEx at execution time (refer to Appendix D: NetEx Default File Configuration for definitions and defaults):

- comapimode
- keyin
- realtime
- ugate_ipaddr
- ugate_port

Step 3. Build an NCT

The PAM file must be built for H300e. Make any necessary changes to the NCT. The source for the NCT is compatible between H300e and H300IPC, however *<u>the PAMFILE must be rebuilt</u>*. As distributed, H300e does not provide an NCT for your site.

Use the system editor to build a network configuration description file. The network configuration describes the topology of the network. The text file consists of configuration statements describing the network. Refer to the *C Configuration Manager and NetEx Alternate Path Retry (APR) User Guide* for information on creating this file.

1. Invoke the Configuration Manager (CM) to process the configuration file.

   There are two major components to CM: the NCT preprocessor and the PAM file generator. The NCT preprocessor reads the configuration text file and transforms this file into an internal data structure called the NCT. The second component creates a binary file (the PAM file) based on user commands and the NCT. The data in the PAM file must then be transferred to NetEx so it can be used by the NetEx routing facility while NetEx applications are being run.

   The CM utility is interactive and various commands may be invoked to generate the configuration that the user desires. Refer to the *C Configuration Manager and NetEx Alternate Path Retry (APR) User Guide* for more information on using this utility. A CM HELP command is also available to assist in the operation of this utility.

2. The following sample command sequence shows how to create a PAM file:

```
@cat,p  NETEX*NTXD-PAM.,///20          (You may supply your own name)
@USE   PAMFILE., NETEX*NTXD-PAM.
@XQT   EXEC.CM          (press enter twice)
        nct      CONFIG.NCTSOURCE       (config name and your NCT source)
        select *
        makepam  netex-hostname  pamfile
        exit
```

Step 4. Create NetEx/IP addressing information

There are four methods of creating/updating the IP information on your system or network to allow for NetEx/IP to operate properly.

1. **Update DNS nameserver information.**

   This method requires that you update the relevant DNS lookup tables with the IP addresses and IP hostnames that will be associated with the HYPERChannel *toGNA* addresses. The IP hostnames **must** be in the following format (case is important):

   `NTX0000`***`uuss`***

   Where ***uu*** is the NETADDR defined for the host in the NCT and ***ss*** is the SMGDREF.

2. **Use the *SET IPROUTE* command in the NTX$INIT file.**

   The third option is to update the NetEx/IP network information via the new NTXOPER command **SET IPROUTE**. Please refer to 'Chapter 6: Operator Interface' for a description of the command. These commands may be placed in the NTXDEFAULT file after the 'ENDINI' statement so that they are executed on every startup.

3. **Use the SET IPROUTE command**

   The last option is to update the NetEx/IP network information via the new NTXOPER command **SET IPROUTE**. Please refer to the 'Chapter 6: Operator Interface' section of the *"NetEx/IP for UNIX or Windows Systems"* manual for a description of the command.

Note: NESi recommends that customers use options #1 or #2 when updating the IP addressing. Updates made via option #3 are only valid for as long as NetEx/IP is running. IP routing information entered with the 'set iproute' command is lost upon recycle of NetEx/IP or dynamic reload of NCT via the 'load nct' NTXOPER command.

Dynamic mapping of GNA addresses to IP addresses is performed during NetEx/IP initialization (and when the LOAD NCT command is issued). The IP addresses that are returned from DNS are saved in an internal NetEx/IP table.

These addresses comprise the HYPERChannel ***toGNA*** addresses, as defined in the NetEx/IP NCT by the NETADDR *(uu)* and SMGDREF (*ss*) parameters.

Step 5. Build CCTABLE (optional)

The code conversion table can now be modified without the need for the product source code or a recompilation of the software.

To create a customized character conversion table, allocate a new file on your OS2200 system, and assign a use name to the file.

```
@cat,p netex*newconv.,///20
@use   cctable.,netex*newconv.
```

Create the source for this table, by supplying a parameter of your use name:

```
@xqt  H300e*release704.cctbld  [ENTER]
 cctable    [ENTER]
You will receive a message, CCTABLE file creation complete…
```

The NetEx run proc should be updated to include:

```
@use  cctab.,netex*newconv
```

The element NTXDEFAULT in your CONFIG file should be updated to include:

```
cctable    cctab
```

The netex*newconv file may be edited and any local changed to the character conversion tables can be supplied. NetEx reads this file at initialization time. If an alternate code conversion table does not exist, NetEx/IP will use the built-in default table.

Step 6. Start NetEx

Start the NetEx run proc.

If you wish to run NTXOPER commands from the console and see the output on the system log, you should start consolerun. NTXOPER can also run from a telnet terminal window on your OS2200 system. In this case all input and output is directed to the terminal window only.

Appendix D: NetEx Default File Configuration

Edit the NetEx DEFAULT file

The NTXDEFAULT file contains default values for NetEx/IP parameters.

Edit this file with the following recommendations:

1. The entries that must be supplied or modified by the customer prior to starting NetEx: NetEx MAY not run if these are commented out of the NTXDEFAULT file:

    a. The first entry (local) in the file defines the name of your local host. Edit this entry to reflect the name of your system as it appears in the Network Configuration Table (NCT).
    b. For the "device1, device2, etc entries in the table the user will need to add these parameters to their NTXDEFAULT.
    c. Ip1intrfc must be set to 0 for Unisys
    d. Comapimode defaults to A, if you use another COMMAPI you will need to change it
    e. Keyin defaults to NETEX
    f. Realtime defaults to 20 (adjust as necessary)
    g. Ugate_ipaddr (refer to table for info)
    h. Ugate_port (refer to table for info)

2. Edit or modify any other parameters for your host and site. The following table lists all of the parameters and their default values.

Lines proceeded by an '*', '#', '"', '!', '.'or '/' are comments in the distributed file, these comments indicate the use of provided program defaults. To override these default values you should duplicate the entry and uncomment, remove the *, and supply your override value. Note that parameters for Protocol Type 4 should not be altered unless directed by NetEx Support, as this protocol is currently not supported in some products.

NTX_Parameter	Default	Definition
ENDINI		In the default file, indicates the end of parameters. Anything after this line will be processed as an NTXOPER command.
local	LOCAL	This is the NetEx name of your host. The local host name is the same name defined on the HOST statement in the NCT and during the MAKEPAM phase when building the PAM file.

NTX_Parameter	Default	Definition
device1	UDP-6950-00000000	The device name consists of the IP protocol, port and the GNA for this IP device interface. • UDP is the only valid IP protocol • 6950 is the port used for this NetEx network 00000000 should be the GNA (NETADDR and SMGDREF specified in the NCT)
device2	(blank – this indicates other interfaces do not exist)	This is the device name of the second network device interface.
device3	(blank – this indicates other interfaces do not exist)	This is the device name of the third network device interface.
device4	(blank – this indicates other interfaces do not exist)	This is the device name of the fourth network device interface.
Local Paths		
cctable	cctable	This is the name of a site modified NetEx code conversion table.
mbxname	mbxname	Ignored in H300e
pamfile	pamfile	@USE name that points to the pamfile (output of the Configuration Manager)
trapcmd	trapcmd	Not supported at this time
Logging Parameters		
logerrors	default	The two allowed values are “default” and “all”: DEFAULT: This value logs adapter and communications errors that are uncommon. Some of these errors are normal but can occur only once every few hours. ALL: This value logs all adapter and communications errors.

NTX_Parameter	Default	Definition
msglvl	interesting	This is the value that controls the verbosity of the message displays. A value must be specified as one of the following: immediate important interesting moderate monitor debug blither A value of 'important' (or lower) must be specified to enable NetEx/IP session establishment messages to be recorded in the 'ntxlog' file.
msgsyslog	1	0- Send minimal output to the NetEx log. 1- Send NetEx messages to the NetEx log
ntxlogname	ntxlog	@USE name that points to the NetEx logfile
numlogs	5	Ignored in H300e
NetEx Parameters		
ackcredit	2	This is the number of buffers that NetEx/IP sends without returning an explicit ack. **Note:** More buffers may be in the process of being sent. Increasing this value has little effect on system load and may decrease NetEx/IP throughput.
bufcnt	2000	This is the number of "segment" sized buffers available for all input and output. **Note:** This parameter should be increased to match the maximum amount of data expected to be in transit (being sent but not yet acknowledged) divided by "segment".
commipod	300	Percentage factor of deadtime used to determine communications interrupted time.
connto	30	This is the initial value for CONNECT TIMEOUT
datato	30	This is the initial value for DATA TIMEOUT. In a heavily loaded system or during transfers to tape, this value should be increased to about 1000.
deadto	60	This is the initial value for DEAD TIMEOUT.
defblkin	32768	This is the default value for NRBBLKIN, if the value zero is specified.

NTX_Parameter	Default	Definition
defblkout	32768	This is the default value for NRBBLKOUT, if the value zero is specified.
dreadqueue	12	This is the number of reads that NetEx/IP has outstanding to the DRIVER. This value may need to be increased, if many very small transfers are needed.
dynpam	0	0 - netex is not using dynamic pams (NCT is necessary) 1 - netex is using dynamic pams (NCT not necessary - only supported with HyperIP)
idleto	6	This is the initial value for IDLE TIME. In networks with many errors or contention, it may help to drop this value to 2 or 3.
ip1intrf	1	If true (1), one IP socket/device for all interfaces. If false (0), a socket/device for each IP interface.
maxblkin	65400	This is the maximum NRBBLKIN value in bytes.
maxblkout	65400	This is the maximum NRBBLKOUT value in bytes.
maxmbxxfer	32768	This is the maximum block that can be sent across the STREAMS pipe facility.
mbufin	8	This is the number of input blocks that NetEx/IP will allow to be outstanding for each user. This may have to be increased for long telecommunications delays.
mbufout	8	This is the number of output blocks that NetEx/IP will allow to be outstanding for each user. This may have to be increased for long telecommunications delays.
multihost	Off	Specifies whether multihost is enabled or disabled. This parameter is important when using TNP.
sndgrnm	On	Specifies if a groupname or a hostname is sent in the connection protocol. This is only pertinent when application specifies a host group for the NetEx Host name. Refer to the CM.

NTX_Parameter	Default	Definition
nodns	0	Enable or disable DNS lookups when the PAM file is loaded. Accepted values are “1” and “0”. The default is 0 (false). When set to **1** (true), NetEx/IP will skip DNS lookups when the PAM file is read. This is only useful if the user intends to ONLY use the “SET IPROUTE” command to manually map “toGNA” addresses to IP addresses.
pollsel	On	This specifies whether or not the dispatcher will ignore wait time calculations less than the highres-timer. Especially useful in order to get good performance with small segments (ON), potentially spending more CPU cycles.
prefprot	2	This value defines the default preferred protocol type to use when connecting to hosts that support multiple NetEx/IP protocol types. Valid values are 2 or 4.
ropclass	G	This defines the default remote operator class for the NetEx/IP operator console. Allowed values are A or G. The default is “A”. “**A**” will allow privileged instructions such as SET, HALT, etc. to be issued via the remote operator facility. “**G**” only allows non-privileged operator commands (DISPLAY, etc.).
segsize	32768	This is the maximum size of a NetEx data block on the network. Value from 256 to 65400.
slim	255	This is the limit on the smax parameter.
smax	32	This is the maximum number of sessions active at any one time. If this value is changed, the file specified by “mbxname” above should be deleted.
smqmax	25	Dictates the maximum session manager request queue depth
timer	2	This is the initial value for the watchdog timer.
usergate	0	Specifies whether NetEx should bring up the TCP XMEM interface – 0 is disabled. (Ignored for H300e)
writeto	45	The number of seconds a write will wait to be accepted by NetEx before failing.
Protocol 2 Throttling		

NTX_Parameter	Default	Definition
defmsecsonewaydelay	0	This is the one-way propagation delay of the network, expressed in milliseconds, and is used if there is no delay value specified in the PAM for this network path. For Type 2 protocol connections, this value represents a fixed propagation delay that never changes. For Type 4 protocol connections, this value represents a starting point that is used for internal bandwidth capacity calculations. However, the delay is continuously measured during each session, and if it changes, the updated value is used for subsequent internal bandwidth capacity calculations.
highresmsecs	10	This is the value of the internal high resolution timer. It is used for internal rate throttling, and should not be changed.
maxkbitspersec	0	This is the maximum rate at which NetEx/IP will deliver data to the network for each network connection. If zero is specified, data will be delivered with no internal throttling. This value is only used for connections to hosts that do not have a ‘rate’ value specified in the PAM. This value is specified in Kbits per second. For example, a value of 50 means 50Kbs; a value of 50000 means 50,000 Kbs (i.e., 50Mbs).
rcvratesecs	2	This is the time interval (in seconds) after which the receive rate for a network connection is recalculated. The receive rate refers to the rate at which the application is receiving data from NetEx.
sndratesecs	4	The actual send rate calculation interval
udpport	6950	Only used with nctless NetEx (not supported at this time), otherwise see device statements
iprecv	250000	IP Receive buffer size
ipsend	250000	IP Send buffer size
ipchksum	1	Not implemented
Protocol 4 Throttling Parameters (May not be supported on all platforms, including Unisys):		
bufolimit	2000	This is the maximum number of outstanding buffers that NetEx will allow per connection.

NTX_Parameter	**Default**	**Definition**
defstartkbitspersec	455000	This is the starting throughput rate at which NetEx/IP will attempt to deliver data to the network for each network connection. If zero is specified, data will be delivered with no internal throttling. This value is only used for connections to hosts that do not have a ‘rate’ value specified in the PAM or NRB. This value is specified in Kbits per second. For example, a value of 50 means 50Kbs; a value of 50000 means 50Mbs (i.e. 50,000 Kbs).
delincp	900	Percentage factor used when decreasing speed because of delay increase.
delincrreq	750	Percentage factor of receive rate used when determining equivalency of increase or decrease of delay.
delincsreq	750	Percentage factor of send rate used when determining equivalency of increase or decrease of delay.
minkbitspersec	0	This is the minimum throughput rate that NetEx will decrease to.
minnpdu	1300	Currently not supported.
modsegsz	0	This indicates whether NetEx may modify the segment size dynamically. This should remain at 0 currently and is not supported.
nakdec	0	This indicates whether NetEx will decrease speed when there are NAKs occurring.
oktodec	1	This indicates whether NetEx is allowed to decrease speed or not.
ratedelaydecp	250 (25.0%)	This is the percentage factor used to decrease the sending rate of a network connection if the round-trip delay increases. This recalculation is performed after the expiration of each interval specified by the ‘rtdelayincsecs’ parameter. The value specified represents a percentage multiplied by a factor of 10.
rateequivph	855 (85.5%)	This is the high bound of the send and receive equivalence adjustment (see ‘rateequivps’). The value specified represents a percentage multiplied by a factor of 10.
rateequivpl	500 (50.0%)	This is the low bound of the send and receive equivalence adjustment (see ‘rateequivps’). The value specified represents a percentage multiplied by a factor of 10.

NTX_Parameter	Default	Definition
rateequivps	750 (75.0%)	This is the percentage factor used to determine equivalence of the send and receive rates for each network connection. These rates are assumed to be equal if they fall within this percentage of each other. This is the initial value used for each network connection, and can be dynamically adjusted based on activity and performance of the network. The value specified represents a percentage multiplied by a factor of 10.
rcvdataqhbytes	20000000	This is the high threshold value (in bytes) for the size of the receiving NetEx DataQue for each network connection. If the size of the Data Queue exceeds this value, subsequent blocks received on the network that are greater than the highest received LRN are NAK'ed and discarded. Subsequent blocks will continue to be discarded until the size of the DataQue is reduced to the value specified by 'rcvdataqlbytes'.
rcvdataqhsegs	15000	This is the high threshold value (in segments) for the size of the receiving NetEx DataQue for each network connection. If the size of the Data Queue exceeds this value, subsequent blocks received on the network that are greater than the highest received LRN are NAK'ed and discarded. Subsequent blocks will continue to be discarded until the size of the DataQue is reduced to the value specified by 'rcvdataqlsegs'.
rcvdataqlbytes	10000000	This is the low threshold value (in bytes) for the size of the receiving NetEx DataQue for each network connection. See the 'rcvdataqhbytes' parameter for the description of how this value is used.
rcvdataqlsegs	8000	This is the low threshold value (in segments) for the size of the receiving NetEx DataQue for each network connection. See the 'rcvdataqhsegs' parameter for the description of how this value is used.
rexmwblks	2	This is the number of blocks to use to calculate the time to wait before rexmitting a block when it is NAKed. Only used if userexmitq is 1.
rtdelayincsecs	60	This is the interval (in seconds) used by the sending side of a network connection after which a check is made for an increase in the round trip delay.
segdownp	300	Currently not supported.
segupp	200	Currently not supported.

NTX_Parameter	Default	Definition
sndrateph	500 (50.0%)	This is the high bound of the ‘sndrateps’ adjustment for each network connection. The value specified represents a percentage multiplied by a factor of 10.
sndratepl	100 (10.0%)	This is the low bound of the ‘sndrateps’ adjustment for each network connection. The value specified represents a percentage multiplied by a factor of 10.
sndrateps	200 (20.0%)	This is the percentage factor used to increase or decrease the send rate for each network connection. The send rate may be increased after the expiration of the interval specified by ‘sndrateincsecs’. The send rate may be decreased after the expiration of the interval specified by ‘sndratedecsecs’. The value specified represents a percentage multiplied by a factor of 10.
sndrateupsecs	20	This is the interval (in seconds) used by the sending side of a network connection after which a check is made for a rate increase or decrease (i.e. the receive rate is greater or less than ‘rateequivps’ of the send rate). If an adjustment is required, the sending rate is increased or decreased by the current value of ‘sndrateps’.
startnpdu	6500	Currently not supported.
startratep	750 (75.0%)	This is the percentage factor used to calculate the initial send rate of each network connection. This value is applied against the maximum rate for the connection, as specified in the PAM, NRB, or by the ‘defstartkbitspersec’ parameter. During the course of the connection, the actual send rate may be adjusted, based on network activity. The value specified represents a percentage multiplied by a factor of 10.
usercvgapq	0	This indicates whether or not NetEx should use the receive gap queue. This should currently remain at 0.
userexmitq	1	Specifies whether or not to use the retransmit queue.
Debug Parameters		
debugdata	0	This is the maximum number of data bytes to trace for any associated data blocks.

NTX_Parameter	Default	Definition
debugmsg	0	This parameter enables or disables the tracing of HYPERchannel messages between NetEx/IP and the network. A value of 0 turns tracing off; any other value turns tracing on.
debugreq	0	This parameter enables or disables the tracing of user requests arriving at the NetEx/IP protocol stack. A value of 0 turns tracing off; any other value turns tracing on. The default is 0.
debugret	0	This parameter enables or disables the tracing of user responses being returned from the NetEx/IP protocol stack. A value of 0 turns tracing off; any other value turns tracing on. The default is 0.
xdbg	0	This is a special debug option to see details of throttling. Setting the value > 0 will turn on xdbg tracing, default is 0.
UDP Transport Parameters		
ipchksum	1	Not implemented
iprecv	250000	IP Receive buffer size
ipsend	250000	IP Send buffer size
mtudisc	0	0 – IP will fragment if necessary (otherwise follow system settings)
udpport	6950	Only used with NCTless NetEx (not supported at this time), otherwise see device statements
UNISYS Configuration Parameters		
comapimode	A	NetEx uses comapi to communicate with cpcom. Any valid mode may be used.
comapibuff	100	Specifies the listen queue size for comapi. This value cannot exceed 20% of COMAPI MAX_UDP_BUFFERS
keyin	NETEX	This is the keyin that ntxoper will respond to when running to the operator console. Any valid keyin may be used
realtime	20	This is the realtime value netex and bfx will use. Any valid realtime value may be used.

NTX_Parameter	**Default**	**Definition**
ugate_ipaddr	127.0.0.1	This is an internal IP address netex will use to communicate with the local applications on this system. If multiple copies of netex are running on the same system, this MUST be a unique address for each copy of NetEx. Any 127. address may be used. Please avoid 127.0.0.0, 127.0.0.1, 127.0.0.254, and 127.0.0.255. This may be used by other applications.
ugate_port	6930	This is a port number to allows netex and the applications to communicate on the same OS2200 system. Port 6930 may be used by multiple copies of NetEx running in the same OS2200 partition. If port 6930 is in use, a new port number maybe used.

Notes:

- Some of the parameters documented above may not be included in the distributed default file. It is the responsibility of the user to enter these values as necessary into the installation-specific copy of the default prior to starting NetEx.

Appendix E: NetEx Default Parameters Mapping

This section maps the NetEx default parameter names with the operator commands and display names. Shaded entries are for NetEx protocol 4 only and are not supported at this time. Blank entries in the table are N/A.

NTXDEFAULT	ntxoper display	ntxoper set	negotiated/ calculated
local	Host		
device1			
device2			
device3			
device4			
logerrors	(not used)		
mbxname			
pamfile		(load nct changes it)	
ntxlogname			
cctable			
msgsyslog	msgsyslog (d p)	msgsyslog	
ropclass	ropclass (d p)	ropclass	
nodns			
ackcredit	Ackcr (d t nref)		
bufcnt			
connto	contime (d p)	contime	
datato	readtime (d p)	readtime	
deadto	deadtime (d p)	deadtime	
defblkin	defbi (d p)	defbi	
defblkout	defbo (d p)	defbo	
dreadqueue	dreadque (d p)		
idleto	idletime (d p)	idletime	
maxblkin	maxbi (d p)	maxbi	Mblki (d t nref) - negotiated
maxblkout	maxbo (d p)	maxbo	Mblko (d t nref) - negotiated
maxmbxxfer			
mbufin	mbufin (d p)	mbufin	Maxrblok (d t nref) - calculated
mbufout	mbufout (d p)	mbufout	Maxtblok (d t nref) - calculated

NTXDEFAULT	**ntxoper display**	**ntxoper set**	**negotiated/ calculated**
smqmax			
segsize	segsize (d p)		Segsize (d t nref) - negotiated
slim	lim ses (d p)		
smax	max ses (d p)	sesmax	
timer	wdogint (d p)	wdogint	
numlogs			
prefprot	prefprot (d p)	prefprot	Prot (d t nref) - negotiated
dynpam			
msglvl	msglvl (d p)	msglvl	
multihost	multihost (d p)	multihost	
sndgrnm			
pollsel	pollsel (d p)	pollsel	
userexmitq	userexmitq (d p)	userexmitq	UseReXmQ (d t nref)
rexmwblks	rexmwblks (d p)	rexmwblks	ReXmWBlks (d t nref)
usercvgapq	usercvgapq (d p)	usercvgapq	UseRcvGapQ (d t nref)
writeto			
xdbg	xdbg (d p)	xdbg	
maxkbitspersec	maxkbs (d p)	maxkbs	Mrate (d t nref) - calculated
highresmsecs			
rcvratesecs			
sndratesecs			
defmsecsonewaydela			
debugreq	dbgreq (d p)	dbgreq	
debugret	dbgret (d p)	dbgret	
debugmsg	dbgmsg (d p)	dbgmsg	
debugdata	dbgdata (d p)	dbgdata	
udpport			
iprecv			
ipsend			
mtudisc	mtudisc (d p)		
ipchksum			
ip1intrf			

NTXDEFAULT	ntxoper display	ntxoper set	negotiated/ calculated
usergate			
rcvdataqlbytes	dtql (d p)	rcvdataqlb	DtQLB (d t nref)
rcvdataqlsegs	dtqls (d p)	rcvdataqls	DtQLS (d t nref)
rcvdataqhbytes	dtqh (d p)	rcvdataqhb	DtQHB (d t nref)
rcvdataqhsegs	dtqhs (d p)	rcvdataqhs	DtQHS (d t nref)
startratep			
defstartkbitspersec			
rtdelayincsecs			
ratedelaydecp			
sndrateupsecs			
sndratepl			
sndrateps			
sndrateph			
rateequivpl			
rateequivps			
rateequivph			
minkbitspersec			
bufolimit	bufolim (d p)	bufolim	BufOLim (d t nref)
oktodec			
nakdec			
modsegsz			
minnpdu			
startnpdu			
segdownp			
segupp			
delincp			
delincsreq			
delincrreq			
commipod			
userexmitq	userexmitq (d p)	userexmitq	UseReXmQ (d t nref)
rexmwblks	rexmwblks (d p)	rexmwblks	ReXmWBlks (d t nref)
usercvgap	usercvgapq (d p)	usercvgapq	UseRcvGapQ (d t nref)

NTXDEFAULT	ntxoper display	ntxoper set	negotiated/ calculated
comapimode			
comapibuff	Comapibuff (d p)		
keyin			
realtime			
ugate_ipaddr			
ugate_port			

Appendix F: NetEx Tools

This section documents the NetEx tools shipped with the product.

If you have any questions on running these tools please contact support@netex.com

NTXMGEN

This tool will generate data for testing purposes. It will prompt the user for parameters.

The prompt will look like this:

```
>@xqt netex*NTXD-exec.ntxmgen        <-User must hit enter twice for prompt
>
PROD$CONF=NETEX*NTXD-CFG.
Using netex interface version =1
running realtime 27
Thu Apr  7 16:29:09 2016: NTXMGEN  V 3.2   12/10/12   65535 Max data
ENTER:
#SESS #BLOCKS SIZE  ODATA LOOPS DMODE VALIDATE HOSTNAME OFFRNAME
NNN   NNNNNN  NNNNN NNN   NNNN  HHHH  Y/N      HHHHHHHH OOOOOOOO
>1 100 1000 0 1 0 n downey ntxmeat
1 ses, 100 blocks, 1000 bytes/blk, 0 odata bytes,
```

Where:

#SESS:	The number of concurrent sessions to process. This must be equal to or less than the number of sessions used by NTXMEAT. No default.
#BLOCKS:	The number of blocks of data to generate. No default.
SIZE:	The size of the blocks of data to generate in bytes. No default
ODATA:	The number of bytes of ODATA to generate. Typically, this parameter can be set to 0.
LOOPS:	The number of times to send all of the blocks. No default.
DMODE:	The DATAMODE to use when sending the blocks. No default. (See NRBDMODE)
VALIDATE:	Should the content of each received block be validated. No default.
HOSTNAME:	The NetEx hostname to send to. No default.
OFFRNAME:	The NTXMEAT Offer to connect to send the data. No default.

NTXMEAT

This tool will read data generated by NTXMGEN. It will prompt the user for parameters.

The execution and prompting will look like this:

```
>@xqt netex*NTXD-exec.ntxmeat        <-User must hit enter twice for prompt
>
PROD$CONF=NETEX*NTXD-CFG.
Using netex interface version =1
running realtime 27
Thu Apr  7 15:59:32 2016 NTXMEAT   V3.2   04/18/14   65535 max data
Enter:
#Sessions Validate OffrName HostName
NNN      Y/N      OOOOOOOO HHHHHHHH
> 1 n ntxmeat unid41c
Making 1 offers of ntxmeat , validate 0, hostname UNID41C
Using COMAPI mode A index=1
Thu Apr  7 16:00:28 2016: COffer: Status: 0, Ind: 1, Session: 1 Try: 0 Loop: 1
Thu Apr  7 16:00:30 2016: Session 1: loop 1
  3.5574 Mbits/s,   0.4447 Mbytes/s, 502.0000 OPs/s, 2 Sec, 1004 Blks, 889356 Bytes
^C*EXECUTION TERMINATED*
```

Where:

Sessions:	The number of concurrent sessions to process. This must be equal to or greater than the number of sessions used by NTXMGEN.
Validate:	Should the content of each block be validated. No default.
OffName:	The OffrName NTXGEN will connect to for the test. No default.
Hostname:	The NetEx hostname to receive from. No default.

Running NTXMEAT and NTXMGEN:

1. On the receiving side, execute NTXMEAT (this MUST be started before NTXMGEN).

   When you start the NTXMEAT application, you will be prompted to specify the number of concurrent sessions and whether you want the application to validate those sessions. Enter the values separated by a space character, then hit ENTER.

   You will need to use CTRL-C (or let the offer(s) time out) to stop the NTXMEAT application when your testing is completed.

2. On the sending host, execute NTXMGEN.

   The NTXMGEN application will prompt you to enter a suite of values to use during the test. Enter the values separated by a space character, then hit ENTER.

   For this example, we specified one (1) session of 99995 blocks of 32000 bytes with zero (0) ODATA, one (1) loop and specify zero (0) for the DMODE. There is no validation required.

Once both processes are up and running, on the NTXMGEN side, after specifying the desired parameters and hitting the <Enter> key, you will see:

```
1 ses, 99995 blocks, 32000 bytes/blk, 0 odata bytes,
 1 loops, datamode 0, validate 0, to ntxmeat  at SUNRISE
Connect: Status: 0,Ind: 0, Session: 1 Try: 1
```

On the NTXMEAT side you should see output similar information to:

```
COffer: Status: 0,Ind: 1, Session: 1 Try: 0
```

When each loop completes, the NTXMGEN side will output the stats for the finished loop:

```
Session 1:
325.5054 Mbits/s,  40.6882 Mbytes/s, 1333.3199 OPs/s, 75 Sec, 99999 Blks,
3199872256 Bytes
```

On the NTXEAT side, the output when the test completes is similar to:

```
CDisc: Status: 0,Ind: 0, Session: 1
Session 1:
325.5052 Mbits/s,  40.6881 Mbytes/s, 1333.3199 OPs/s, 75 Sec, 99999 Blks,
3199872256 Bytes
```

Index