H897IPV NetEx/IP® Requester

for Stratus OpenVOS Operating Systems

Release 2.1

Software Reference Manual

MAN-REF-H897IPV-R2.1-01

Revision Record

Revision	Description
01 (11/2011)	Manual released corresponding to release 2.1.

The U.S. Department of Commerce restricts the distribution of technical information contained in this document when exported outside the U.S. Therefore, careful attention should be given to compliance with all applicable U.S. Export Laws if any part of this document is to be exported.

You may submit written comments using the comment sheet at the back of this manual to:

Network Executive Software, Inc.
Publications Department
6420 Sycamore Lane, Suite 300
Maple Grove, MN 55369
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 Network Executive Software (“NetEx Software”) H897IPV NetEx/IP Requester for the Stratus OpenVOS operating system.

This document contains the following sections:

- “Introduction” on page 1 of this manual is intended for all readers.
- “NetEx Requester Utility Programs” on page 9
- “Appendix A. NetEx Request Block” on page 15
- “Appendix B. NRBSTAT Error Codes” on page 27

Appendices include a list and description of the error messages and codes issued by NetEx.

Readers are not expected to be familiar with NetEx 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-REF-H210IPZ	*H210IPZ NetEx/IP for IBM z/OS Operating Systems Release 7.1 Installation Reference Manual*
MAN-REF-H892TV	*H892TV Print File Transfer (PFX™) for Stratus OpenVOS Systems Software Reference Manual*
MAN-REF-H893V	*H893V USER-Access™ for Stratus™ OpenVOS Systems Software Reference Manual*

Notice to the Reader

The material contained in this publication is for informational purposes only and is subject to change without notice. Network Executive Software is not responsible for the use of any product options or features not described in this publication, and assumes no responsibility for any errors that may appear in this publication. Refer to the revision record (at the beginning of this document) to determine the revision level of this publication.

Network Executive Software does not by publication of the descriptions and technical documentation contained herein, grant a license to make, have made, use, sell, sublicense, or lease any equipment or programs designed or constructed in accordance with this information.

This document may contain references to the trademarks of the following corporations:

Corporation Trademarks and Products

Network Executive Software **NetEx, BFX, PFX, USER-Access, NESiGate**

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

The installation information supplied in this document is intended for use by experienced System Programmers.

Software Modification Policy

Modifications to H897IPV that are not specifically authorized by NetEx Software are prohibited.

Any unauthorized modifications to H897IPV may affect its operation and/or obstruct NetEx Software's ability to diagnose problems and provide corrections. Any work resulting from unauthorized modifications shall be paid by the customer at NetEx Software's then-current support rates and may result in the immediate termination of warranty/support coverage.

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 the adapter or host interface that dynamically converts the host data from one character set to another. An adapter configured with the code conversion has a special 1K RAM that is used for code conversion. This RAM can be loaded with any type of code (for example, ASCII, EBCDIC, et cetera).

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.

link: (1) A joining of any kind of networks. (2) The communications facility used to interconnect two trunks/busses on a 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 on a processor interface board (obsolete). The latter case uses host-resident drivers as interfaces.

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, and et cetera) of a distributed network. OSI was defined by the ISO.

processor interface (PI): A PI interfaces a minicomputer with an adapter. The PI is a board(s) that contains a microprocessor and memory. The processor interface is generally installed in the host. Some types of PIs contain NetEx.

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

Introduction

Network Executive Software® NETwork EXecutive (NetEx®) family of software products allows two or more application programs (which may be on different host computers) to communicate with each other at multi-megabit speeds. The NetEx family of software consists of different versions of NetEx for use with different operating systems. All of these versions provide a common high-level interface to simplify programming requirements. Network Executive Software has utility programs available for use with NetEx, such as USER-Access™, Bulk File Transfer (BFX™), and Print File Transfer (PFX™) utilities.

H897IPV is a portable implementation of one variety of NetEx interface software. H897IPV is designed to use standard Stratus OpenVOS communication facilities to provide a remote procedure call interface to H210IPZ's "TCP NetEx/IP Proxy" (TNP) component. In this implementation, the NetEx protocol software is located in the NetEx software on the z/OS host. H897IPV resides on the Stratus host and is the user interface to H210IPZ. H897IPV passes NetEx requests between the Stratus host and H210IPZ on the z/OS host across the network using TCP/IP.

This manual describes NetEx and H897IPV in general terms. For greater detail, please consult "Reference Material" on page iv.

Figure 1 below shows an example of an H897IPV/H210IPZ TNP configuration.

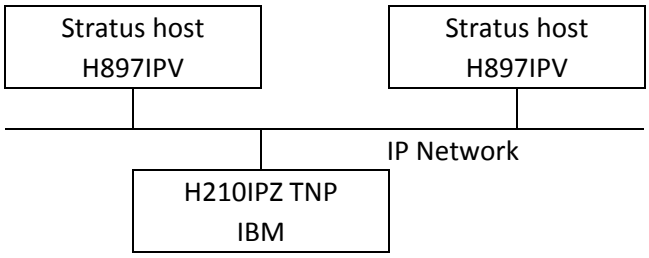


Figure 1. Network Configuration Example

The following sections describe the characteristics of NetEx and how NetEx uses the International Standards Organization guidelines for open systems interconnection:

- "NetEx Characteristics"
- "Host Based NetEx"
- "CP NetEx"
- "NetEx and the ISO Model"
- "Session Layer Services"
- "Transport Layer Services"
- "Network Layer Services"
- "Driver Sublayer Services"

NetEx Characteristics

NetEx centralizes network considerations for networks such as FDDI, Ethernet or other supported networks into two pieces of software. The following paragraphs describe the characteristics of the NetEx software including the external interface, internal interaction, NetEx connections, design efficiency and flexibility, block segmenting, Alternate Path Retry, remote operator interface, and the basic I/O flow.

External Interface

The NetEx external interface for the application programmer is common for all versions of NetEx. NetEx provides requests for use in the programs that call NetEx. These calling programs may be written in the C language. NetEx 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 also provides an operator interface that monitors and controls certain NetEx functions.

Internal Interaction

The internal operation of all supported versions of NetEx is consistent and allows the different versions to interact freely. Thus any program using NetEx may communicate with any other program on the network that is also using NetEx.

To facilitate communication between hosts of different manufacture, NetEx supports code conversion. NetEx can call on supported network adapters to do code conversions and data assembly/disassembly, even if the sending adapter has that option installed.

NetEx Connections

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

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

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

Once NetEx accepts data from the caller, it is responsible for delivering the data to its destination. The NetEx subsystem on each host handles flow control, error recovery, and any other special considerations such as satellite links.

NetEx optimizes data transfer throughput using a high degree of parallelism. That is, under normal circumstances, simultaneous supported network adapter I/O, NetEx 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 multimegabit range).

Block Segmenting

NetEx products provide block segmenting at the transport layer. NetEx 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. This segmenting is transparent to the session user but provides control of the transmitted block segment size. This is especially useful for satellite communication.

Remote Operator Interface

This version of NetEx provides a remote operator interface that allows users to issue NetEx operator commands to other defined NetEx hosts on the network. Other users may also be the remote operator for this NetEx. Security features are provided.

Basic I/O Flow

Figure 2 below shows the basic I/O flow between two programs using host based NetEx. The calling program communicates with NetEx through the NetEx user interface. NetEx then uses the Network Executive Software hardware to communicate with the calling program on the other processor.

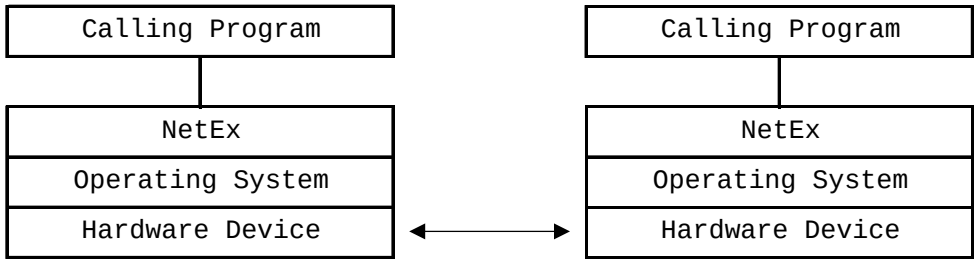


Figure 2. Basic I/O Flow

Host Based NetEx

Host based NetEx is an architecture that is designed for implementation on very large mainframe computers or on some smaller machines that cannot support the creation of a standard Network Executive Software driver product. A 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.

Host based NetEx products are implemented in Assembler, PASCAL, and other languages.

CP NetEx

Coprocessor (CP) NetEx is very similar to host based NetEx, but is inside the Network Executive Software adapter. CP NetEx uses the processing capability of the PI, thus eliminating the drain on the host storage and processing capabilities. A CP NetEx driver resides in the host to provide a programming interface to the CP NetEx.

IP NETEX

This is the new generation of host-based NETEX products that provide support for standard IP networks, rather than HYPERchannel trunk and HCM/FDDI media only. These products are an extension of and retain all the features available in the previous generation of the host-based NETEX model.

NetEx and the ISO Model

In creating NetEx, 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, networks, etcetera 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.

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-to-NetEx 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

Table 1. ISO Model

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

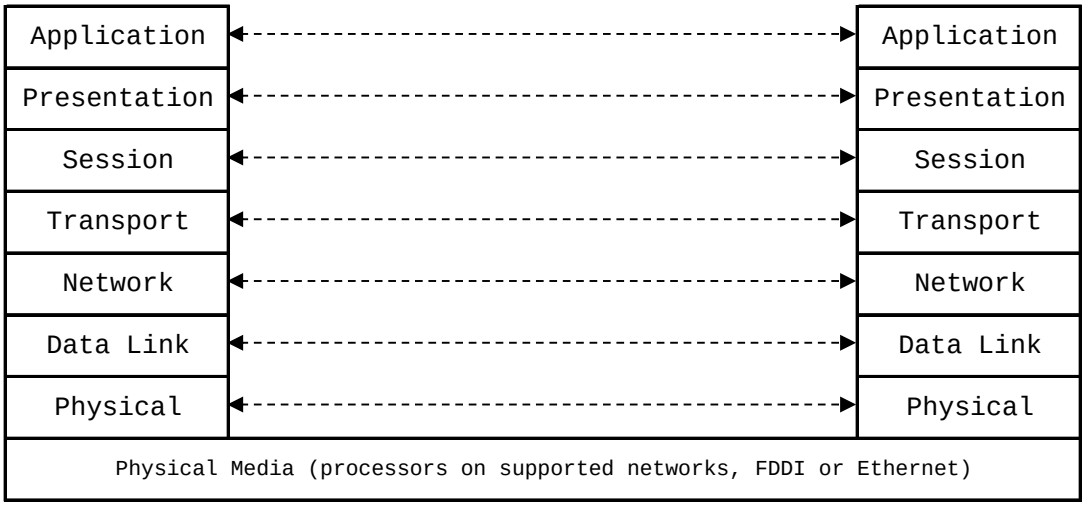


Figure 3. ISO Model Communication

Notice that the corresponding layers appear to communicate directly as indicated by the dotted lines, 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 4 shows that the host's supported network interface hardware and firmware form the lower two layers. The NetEx software comprises the next three layers and provides complete session, transport, and network layer interfaces. This leaves the user free to simply write the application programs that use NetEx or to use Network Executive Software utilities.

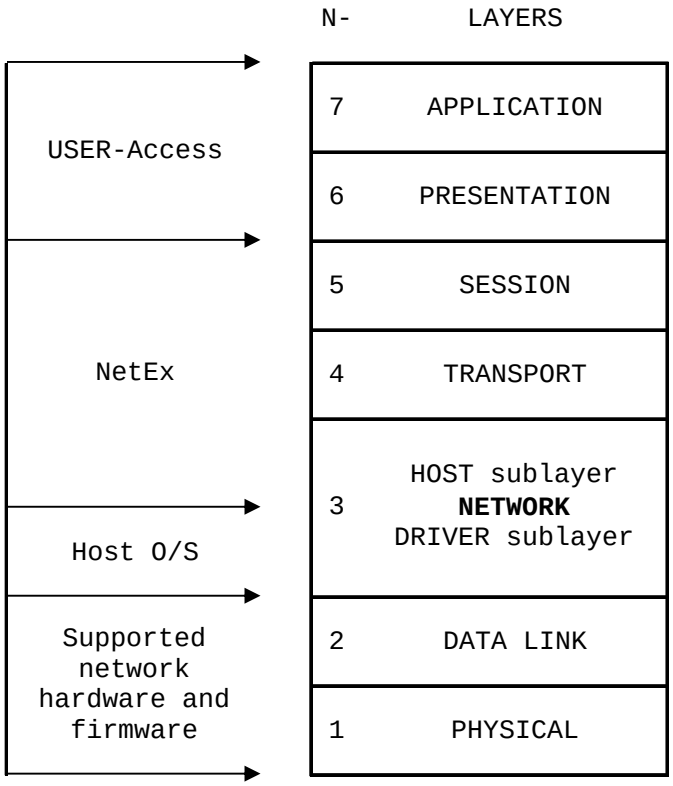


Figure 4. NetEx and the ISO Model

Session Layer Services

As the highest layer within NetEx (referring to the ISO model in Figure 4), the session layer software provides the general interface to the user's application/utility program. The NetEx 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 Request Block (NRB) (containing parameters) and the NetEx requests. 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. 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 is responsible for managing the network path chosen by the session software. This means that the session user need not be concerned with the actual hardware and software used to transmit data, nor with NetEx-to-NetEx message delivery. The transport layer sets up tables, provides buffering, and establishes linkages to manage the flow of information. In addition, 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 communications link as busy as possible and assures timely arrival of data to the user.

Network Layer Services

The network layer software provides link independence for the higher layers of NetEx 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 proper to route the data through the network. If the protocol information overflows the supported network message proper, 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. The driver converts network sublayer I/O for a particular network path into a form that is understandable to the devices. The driver is responsible for delivering and receiving 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 provided by the adapter type and requested by the user's data mode parameter.

NetEx Requester Utility Programs

The following utility programs are provided with the H897IPV product:

ntxoper This is a NetEx operator program which can be used to examine the NetEx parameters of any host on the network which supports the NetEx operator interface.

ntxverify This is a test program which provides a very simple test of connectivity from the application to the H210IPZ TNP where the local interface is connected. See "Step 8. Perform additional verification testing" in the *H897IPV NetEx/IP® Requester for Stratus OpenVOS Operating Systems Memo to Users* (MTU-H897IPV-R2.1) document for more information.

ntxread This is one of the programs used to show the C programmer interface.

ntxwrite This is one of the programs used to show the C programmer interface.

The following pages describe the ntxoper utility program in more detail.

NetEx Requester Operator Utility

Overview

The operator interface is designed to allow limited manual control and display of NetEx resources such as remote supported Network Adapters, remote hosts, or particular types of NetEx services. The NetEx operator facility accepts commands interactively.

NTXOPER - NetEx Requester Operator Interface

NTXOPER is a NetEx local/remote operator program for accessing NetEx on any network host capable of supporting the NetEx remote operator interface. It can establish a connection to a supporting NetEx on either the local or a remote host, passing commands to it and displaying response data.

The NTXOPER program operates either in local or remote mode. When the program is initiated, it automatically comes up in local mode. That is, all commands are sent to the H210IPZ TNP. In remote mode, all such commands are sent to the specified remote NetEx host.

The NetEx remote operator display commands are somewhat host specific: see the appropriate NetEx manual.

Commands

NTXOPER commands and parameters may be entered either from the NTXOPER command line interactively or using the NTXOPER command prompt. Issuing commands from the command line is done in the normal manner. The NTXOPER utility is usually located in *>nsc>ntxr>bin*. For example:

```
ntxoper [command] [operatorcommand]
```

To issue commands interactively, you must enter the following command:

```
ntxoper
```

The program will respond with the *ntxoper(host)>* prompt, after which commands may be entered. For example:

```
ntxoper(VOSTNP)> operatorcommand1
ntxoper(VOSTNP)> operatorcommand2
```

NTXOPER commands may be entered either in upper or lowercase.

BYE, EXIT, and QUIT Commands

The BYE, EXIT, and QUIT commands are used to exit the NTXOPER program. These commands have the following format.

Command (Select One)	Parameters
BYE EXit Quit	

BYE This is a verb for this command.

EXit This is a verb for this command.

Quit This is a verb for this command.

HELP Command

The HELP command provides a brief description of all valid NTXOPER commands.

The HELP command has the following format.

Command	Parameters
Help	

Help This is the verb for this command. See the display example in Figure 5 on page 12.

LOCAL Command

The LOCAL command is used to select the local mode and/or provide operator command information. In local mode, commands other than NTXOPER commands are sent to the H210IPZ TNP NetEx for interpretation. This is the initial mode of the NTXOPER program. If multiple TNP units are defined in the local DXNRL configuration file, the LOCAL command defaults to accessing the first unit defined in the file.

The LOCAL command has the following format.

Command	Optional Parameter
LOCal	HELP

LOCal This is the verb for this command. Issued by itself, LOCAL selects the local mode.

HELP This optional parameter lists all valid operator commands for the local NetEx host.

REMOTE Command

The REMOTE command is used to select the remote mode and/or provide operator command information. In remote mode, commands other than NTXOPER commands are sent to the specified remote NetEx host for interpretation.

The REMOTE command has the following format.

Command	Required Parameter	Optional Parameter
REMote	hostname	HELP

REMote This is the verb for this command.

Hostname This required parameter is the remote host name identifier.

HELP This optional parameter specifies that all valid operator commands for the remote NetEx host should be listed.

Examples

Commands which are not valid NTXOPER commands are automatically sent to the currently selected host NetEx for interpretation. In this way, remote operator commands may be entered directly on the NTXOPER command line. For example, a display of the active sessions on a remote NetEx whose host name is PDP may be obtained with the following NTXOPER command sequence:

```
ntxoper
ntxoper(VOSTNP)> REMote pdp
ntxoper(PDP)> d s
ntxoper(PDP)> EXit
```

The first command in this sequence (*ntxoper*) enters the NTXOPER utility program. Next, *REMOTE PDP* selects the remote host PDP. This is a valid NTXOPER command and thus, is interpreted by NTXOPER. *D S*, or Display Sessions, is passed by NTXOPER to the NetEx remote operator interface on the PDP host. Host PDP would then return a list of all currently active sessions to NTXOPER for display. Finally, the *EXIT* command terminates the NTXOPER program.

Figure 5 through Figure 7 show example program output from each of the following commands.

```
HELP
REMOTE PDP HELP
DISPLAY SESSIONS
```

In each case, the host from which the commands were issued was VOSTNP.

```
   ntxoper
 Current hostname:  VOSTNP
 NTXOPER version x.x interactive local/remote NetEx operator

ntxoper(VOSTNP)> help
NTXOPER            version x.x interactive local/remote NetEx operator
                   (uppercase denotes required characters)
BYE                -- same as EXit
Exit               exit ntxoper
Help               displays this message
LOCal              switch to local command mode
LOCal HELP         display local NetEx operator commands
Quit               -- same as exit
REMote name        switch to remote command mode -host 'name'
REMote name HELP   display remote NetEx operator commands
![args]            invokes the VOS shell
?                  get help from local NetEx

ntxoper(VOSTNP)> exit
```

Figure 5. Screen Display: HELP Command (NTXOPER Utility)

```
   ntxoper
 Current hostname:  VOSTNP
 NTXOPER version x.x interactive local/remote NetEx operator

ntxoper(VOSTNP)> remote pdp help
NetEx response from host: PDP
Display /Host     Level      Memory       Parms      Unit
      Session     Transpo    Network      Driver     Adapstat
      Ipistat     HITrace    NHITrace     HOTrace    NHOTrace
      AITrace     AOTrace    EXloss       MBIn       MBout/
Set Parmname value (use DIS PARMS for list of parm names)
Halt /Sref Tref/ ref#
Halt ALL

ntxoper(PDP)> exit
```

Figure 6. Screen Display: REMOTE PDP HELP Command (NTXOPER Utility)

```
   ntxoper
 current hostname:  VOSTNP
 NTXOPER version xx interactive local/remote NetEx operator

ntxoper(VOSTNP)> remote vax
ntxoper(VAX)> d s

NetEx response from host: VAX
Host VAX      Active Sessions
 Sref  Task ID  Tref   State      Name     Host    Rnref Msg In Msg Out
----- -------- ----- ---------- -------- -------- ----- ------ -------
   3         0     3 DATA                VAXIPI       4      2       1
   NA        0    NA OFFERED    CPRNCT00
   NA        0    NA OFFERED    CPBM0000
   NA        0    NA OFFERED    CPLPBR00
   NA        0    NA OFFERED    NTXNCTL0
   NA    20001    NA OFFERED    BFXJS

ntxoper(VAX)> exit
```

Figure 7. Screen Display: DISPLAY SESSION Command (NTXOPER Utility)

Appendix A. NetEx Request Block

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

Each time a program makes a request to NetEx, the program specifies an NRB to be associated with the request. NetEx passes status information about that request back to the program via the NRB. Therefore, only one NetEx request may use an NRB at one time. If concurrent read and write requests are used, or if a server program is to 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 somewhat portable between machines.

- Before initiating a connection, it is the user's responsibility to clear the NRB, including the OS dependent portion, to zeroes. When the connection is initiated, the user places whichever non-default values are needed in the user part of the NRB, and invokes NetEx service. Once the connection is initiated, the user must not change the OS dependent part of the NRB between calls to NetEx.
- 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 the entire 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 will detect the condition and handle that 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 CDC CYBER has an addressable unit of 60 bits, Unisys Computer Systems 1100 is 36 bits, IBM and Digital are generally 8 bits, Stratus is 8 bits, HP9000 is 8 bits, etcetera. Almost all UNIX systems are 8 bits.

NRB Fields

Figure 8 on page 16 shows the fields in the NRB. Most NRB fields are one word long, (32 bits). The NRB contains forty words (160 bytes).

Many of the NRB fields are or could be updated by either the program or NetEx 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 as their values change. These fields are initially specified during the OFFER and CONNECT requests. When the offering task receives the connect, the negotiated values are set in the offered NRB. When the SCONFIRM is sent, the negotiated values are set in the NRB associated with the read of the SCONFIRM information. Following attempts to change these fields will have no effect.

C programs should define and reference NRBs using the structure definitions shown in Figure 8 on page 16. Alternately, an NRB may be defined as an array of 40 integers. In this case, the NRB fields must be referenced using the index values shown on the left side of Figure 9 on page 17.

```
typedef int int32;


typedef int32 osdep[l6];


struct nrb { int32 nrbstat;
int32 nrbind;
int32 nrblen;
int32 nrbubit;
int32 nrbreq;
int32 nrbnref;
int32 nrbnref;
char *nrbbufa;
int32 nrbbufl;
int32 nrbdmode;
int32 nrbtime;
int32 nrbclass;
int32 nrbmaxrt;
int32 nrbblki;
int32 nrbblko;
char *nrbprota;
int32 nrbprotl;
char *nrbresv1;
int32 nrbresv2;
char nrboffer[8];
char nrbhost[8];
int32 nrbresv3;
int32 nrbuser;
osdep nrbosdep;
};
typedef struct nrb NRB;
```

Figure 8. Structure Definitions

0	NRBSTAT - NETEX request status returned to user
1	NRBIND - Data type indication from OFFER/READ
2	NRBLEN - Length of data
3	NRBUBIT - Unused bit count
4	NRBREQ - User request code
5	NRBNREF - NETEX reference number identifying connection
6	NRBBUFA - Starting address of user's buffer
7	NRBBUFL - Length of user's buffer
8	NRBDMODE - Datamode for WRITE request
9	NRBTIME - Request timeout in seconds
10	NRBCLASS - Class of service
11	NRBMAXRT - Maximum data rate permitted
12	NRBBLKI - Maximum buffer size for input requests
13	NRBBLKO - Maximum buffer size for output requests
14	NRBPROTA - Address of Odata
15	NRBPROTL - Length of Odata
16	NRBRESV1 - Reserved
17	NRBRESV2 - Reserved
18	NRBOFFER - (Session Level) Offer name (8 bytes)
20	NRBHOST - (Session Level) Remote hostname (8 bytes)
22	NRBRESV3 – Reserved
23	NRBUSER – Reserved
24-39	NRBOSDEP - Reserved (operating system dependent data)

Figure 9. NetEx Request Block (NRB) Fields

The following paragraphs describe the fields in the NRB shown in Figure 9.

NRBSTAT

NRBSTAT contains a 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, then NRBSTAT is 0. If the request was unsuccessful (NetEx or the service routine detected an error), NRBSTAT contains a binary representation of a decimal error code. The meanings of the error codes are specified in the NRBSTAT Error Codes section of the *PDNT3/NDNT3 DX NetEx Coprocessor Customer Software Reference Manual*.

The implementation user interface must be constructed so that a program polling NRBSTAT (to determine if the request was successful) immediately 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 service call is issued, and NETEX 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 (!)	Connect Indication
INDCONFIRM (2)	Confirm Indication
INDDATA (3)	Normal data
INDEXPDATA (4)	Indication Reserved
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, 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.
- If the data is “damaged” in input (for example, user buffer too small) then NRBIND will reflect the type of data received.

NRBLEN and NRBUBIT

NRBLEN and NRBUBIT together define the amount of useful data for input and output NRBLEN specifies the number of bytes that are needed to contain the data. NRBUBIT specifies the number of bits in the last bytes 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 CDC CYBER computer wished to send exactly 35 of its 60-bit CM words to an IBM processor and wished it returned later. The CYBER user would specify NRBLEN=35 and NRBUBIT=O. Datamode would be bit stream. NetEx would record that 35*60=2100 bits of information was sent over the network. The IBM user would receive the information with NRBLEN=263 (bytes) (8*263=2104 bits) and NRBUBIT=4 (bits). The IBM user could later specify the same length parameters on output and return precisely 35 CM words back to the CYBER.

A second example involving character conversion: Suppose a CRAY wished to send 151 ASCII characters (8*151=1208 bits) to a Unisys and have them converted to Field-data. The CRAY user could specify NRBLEN=19 (64 bit words) (64*19=1216 bits) and NRBUBIT=8 (bits) since 7 characters will be in the trailing CRAY word. The CRAY driver would send the ASCII over the network and record that 8*151=1208 bits of information were sent. The Unisys would select sixth-word AID mode and code conversion and determine that (1208/8)*6=906 bits of information would result. It would report to the Unisys caller that NRBLEN=26 (36*26=936) and NRBUBIT=30 so a single character will be found in the last of the 26 Unisys words.

Note that 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.

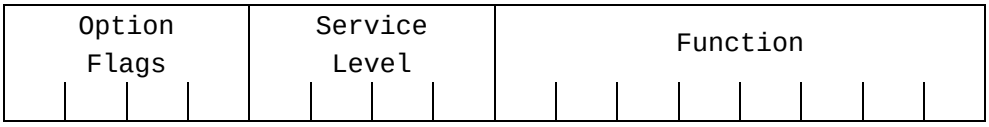
Transmitting or receiving zero-length information is possible. Zero-length data is treated as a separate transmission and is received at the other end in chronological order (as is any other data). On both the transmit and receive sides, NRBLEN will be set to zero.

If NRBUBIT is non-zero, the unused bits are not set to zero or any other value by NetEx. The calling program must handle any 'garbage' that may be placed in the last word of the transfer.

NRBREQ

NRBREQ is the request code that is to be given to NetEx. This is a 16-bit binary value that contains the type of request (example: SREAD) that NetEx is to perform.

NRBREQ has the following format:



Option Flags

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

0xxx	Normal processing. NetEx will return control to the caller when NetEx has internally queued the request.
1xxx to 7xxx	Reserved
8xxx	WAIT. NetEx or the NetEx user interface is not to return control to the user program until the request is complete
9xxx to Fxxx	Reserved

Service Level

Indicates if the request is a SESSION, TRANSPORT, NETWORK, or DRIVER type of request. The values are assigned (in hexadecimal) as follows:

x0xx	- Session request
x1xx	- Transport request
x2xx	- Network request
x3xx	- Driver request
x4xx to xExx	- Reserved
xFxx	- Reserved (effects Function values)

Function

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

xx0l	- Connect (valid for S, T, N, and D levels)
xx02	- Confirm (valid for S, T, N, and D levels)
xx03	- Write (valid for S, T, N, and D levels)
xx04	- Reserved
xx05	- Close (valid for S and T levels)
xx06	- Disconnect (valid for S, T, N, and D levels)
xx07 to xx80	- Reserved
xx81	- Offer (valid for S, T, and N levels)
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 an SREAD with wait processing. Wait processing is 8xxx, SREAD is a x0xx Service Level plus a xx82 Function. This totals an 8082 (hexadecimal) request code.

NRBNREF

NRBNREF is the 16-bit, internal NetEx identifier that distinguishes this connection from all others maintained by this copy of NetEx. When initial connect or offer requests are made at the Driver, Network, or Transport levels, this field must be filled in by the caller. If issued at the Session level, this value is assigned by NetEx when a connection is established. Only the lower 16 bits of the NRBNREF field are used. The high order 16 bits must be zero.

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 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 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 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 program on any write-type request (connect, confirm, write, close, or disconnect) that is issued at the session, transport, network, or driver level. NRBDMODE is always specified as a 16-bit quantity. Datamode is forwarded through all layers of NetEx. When the receiving entity receives 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.

Datamode supports two basic modes: manual and auto datamode. Note, however, that this version of NetEx does not support manual.

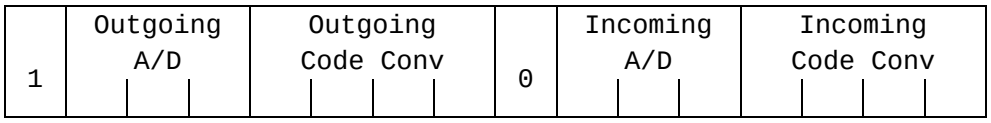
Manual Datamode

Manual datamode allows complete specification of the assembly/disassembly and code conversion functions on both adapter output and adapter input. In the manual datamode, the caller has total control over the adapter facilities. The user must determine which assembly/disassembly modes and code conversion tables are significant to the two adapters involved in the transfer. Refer to the Adapter reference manuals for the Adapter being used.

The datamode field is always in the “datamode” field of the driver protocol information to assist this incoming driver function. When the data is received, each driver calculates the amount of useful information received based on the incoming AID mode specified and passes it up to the user read request. The read NRB will contain exactly the same datamode field as specified by the transmitter when the original message was sent.

Note: Manual Datamode is not available on the H897R interface to NetEx.

Manual datamode has the following format:



'1'

In the high order bit is the manual mode indicator.

Outgoing A/D

Indicates the data assembly/disassembly to be performed on data as it goes out onto the network. This information is added to the "transmit data" function code when the user data goes over the network.

Outgoing Code Conv

Indicates the code conversion to be performed on data as it goes out onto the network. This information is added to the "transmit data" function code when the user data goes over the network. This field is effectively unused since the A400 processor adapter does not support code conversion.

'0'

In the high bit of the low order byte is the low order byte is the manual mode indicator.

Incoming AID

Indicates the data assembly/disassembly to be performed on data as it goes from the network to the receiving program. This information is added to the "input data" function code when the receiving driver gets the message from the network.

Incoming Code Conv

Indicates the code conversion to be performed on data as it goes from the network to the receiving program. This information is added to the "input data" function code when the receiving driver gets the message from the network. This field will only be used if the receiving host-processor adapter supports code conversion.

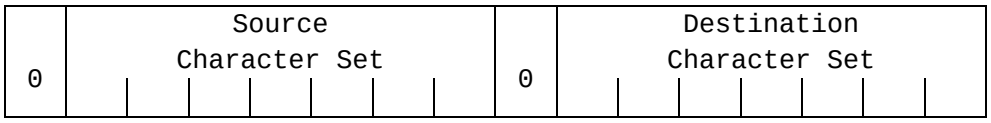
Auto Datamode

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

Auto datamode supports three conversion options.

- Bit Stream - where the bit pattern sent is precisely reproduced in the destination machine.
- Octet - where 8-bit binary quantities are sent from one machine to another, using an 8-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 datamode indicator.

Source Character Set

Indicates the conversion option (from Table 2 below) 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 (from Table 2 below) of the data going to the destination program. For example, a conversion from EBCDIC (3) to ASCII (2) would be entered as the hexadecimal value of 0302 or decimal value of 770.

Indicator	Conversion Option
0	Bit stream mode
1	Octet Mode
2	ASCII (8 bit)
3	EBCDIC
4	Reserved
5	BCD (Honeywell)
6	Field-data (UNISYS)
7	Display code (CDC)

Table 2. Auto Datamode Character Sets

The processing rules for auto datamode are as follows:

- The transmitting driver examines tile source character set specified. The character set implicitly specifies the method used to represent those characters on tile transmitting machine. The driver selects an AID mode so those characters will be each sent over the network as an 8-bit quantity. If the code conversion memory is installed, the transmitting driver will select a code conversion function to hardware convert tile character set before the information is sent over the network. If code conversion is used, the transmitting driver sets source character set to the value of the destination character set in the datamode field of the outgoing message proper.
- The receiving driver always reads a message proper in “octet” mode. By examining the datamode field and determining that character data is being sent, it selects an AID mode to convert the 8-bit quantities coming over tile network to the bit configuration used by the destination character set. If code conversion hardware is installed and the source character set does not equal the destination character set in the message proper, then the data is code converted on input. Following such conversion the source character set field in datamode is set to the destination character set value before the datamode is passed up lo the receiving caller.
- If neither adapter has code conversion, and the character sets in the datamode field are still not equal, then software code conversion is performed and the two fields are set equal.
- If the incoming driver determines that the destination character set is not “significant” on its own host (for example, sending Field-data code to a PDP-11) 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.”
- If the destination character set is a 6-bit code, code conversion hardware is required on the adapter for the 6-bit character machine.

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 NCT.

1 Use Version 1 NetEx protocol. This protocol is used in release I and release 2 NetEx products, but is not supported in Release 3 NetEx.

2 Use Version 2 NetEx protocol. This protocol is used in release 2 and release 3 NetEx products, but is not supported in Release 1 NetEx. This version of NetEx supports class 2 protocol.

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

When an offer or connect completes, the value of this field should contain the protocol version actually negotiated. If Network or Transport services are selected and more than one protocol at this layer is concurrently available, then an installation default is returned. If Session services are requested, then tile default returned may depend on the protocol desired by the remote host as determined in the local Network Configuration Table.

NRBMAXRT

NRBMAXRT (maximum rate) is a connection negotiation parameter that specifies the maximum data rate possible for the connection. NRBMAXRT is used for Session and Transport levels only. This field is for informational purposes only on the session layer.

The NRBMAXRT value is based on the user’s specification or the physical characteristics of the links between the two NetEx calling programs. 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 32-bit positive quantity giving the speed of the link in 1000’s of bits per second. Thus a connection using a terrestrial link adapter whose line speed was generated as 230K bits per second would have 230 placed in this field.

It should also be noted that 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 link 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. (The meaning of NRBBLKI and NRBBLKO is for Network requests is described at the end of this discussion.) This parameter should be provided with the connect or offer request. During the protocol 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 installation systems programmer must supply two values controlling these buffer sizes. 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.

If the connection established is a network or driver level connection then NRBBLKO and NRBBLKI may be adjusted to reflect the maximum size of data block that may be sent as a datagram on the path specified by the connect. If the application negotiated size is smaller than the maximum NPDU size, then the negotiated parameters will be unchanged. If the maximum NPDU size is smaller, then this maximum size will be returned in both NRBBLKI and NRBBLKO.

One default option is available with these fields. If a zero is specified in either one of these fields, then the value used for negotiation will be an installation supplied default that is provided at NetEx installation time. Note that the values implied by 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.

For Network layer requests, the NRBBLKI and NRBBLKO fields are used to inform the Network layer of the maximum amounts of Odata and Pdata that are to be used to send and receive data in this connection. These limits are dependent on many things: the buffer capacities generated in both the local and remote copies of NetEx, and the physical limitations of the media connecting the two hosts. When this NOFFER completes with a Connect Indication, then these fields will have the actual limits for Odata and Pdata size in the connection sent to them. Unlike other layers of NetEx service, tile Network Service will return the maximum that is available if the caller's size request is not available. The caller is responsible for scaling tile buffer sizes downward accordingly.

The maximum size of Pdata is specified in addressable units. The maximum amount of Odata is specified in octets.

NRBPROTA and NRBPROTL

The NRBPROTA and NRBPROTL are connection negotiation parameters that permit the application to provide Odata to the called layer of NetEx. 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." This means it will be represented in the best way that the particular host can handle strings of 8-bit binary quantities. (for example, 1/byte, 4/36-bit word, etcetera)
- The maximum amount of Odata that may be sent is limited. This maximum is installation dependent and may typically be 256 bytes or less. Each version of NetEx will have a generated maximum on the number of bytes of Odata that it is prepared to accept in incoming messages. In the Network, Transport, and Session levels, 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 "fissioning" of network messages, which increases network traffic and decreases network performance, often by a factor of two.

  Note: Not all implementations of NetEx support the use of Odata. No Network Systems utilities use Odata. Consult Network Systems personnel before using Odata to determine if it is available.

On a write-type operation, no Odata will be sent if NRBPROTL is zero. If a nonzero 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.

In all cases, NRBPROTL contains the length of the Odata in octets, not "addressable units."

The protocol field has special significance when used wiU1 Driver level requests, in that the Odata contains the network Message Proper, where the Pdata contains the Associated Data.

NRBRESV1 and NRBRESV2

NRBRESV1 and NRBRESV2 are reserved for possible future NetEx enhancements.

NRBOFFER

NRBOFFER specifies the offered name (the name of the process to be matched when the offer and connect requests meet). Names of all processes are uppercase alphanumeric data that are up to 8 characters in length. Names less than 8 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 significant in ASCII should be used during the name matching process.

NRBHOST

NRBHOST specifies the symbolic name of the host computer that is to be addressed to match an offer request. Names of all hosts are specified by the installation systems programmer. All host names are uppercase alphanumeric data that are up to 8 characters in length. Names less than 8 characters long should be padded with blanks.

NRBRESV3

NRBRESV3 is reserved for possible future NetEx enhancements. Programs should leave binary zeroes in these fields.

NRBUSER

This field is reserved for the user interface. Applications should not use this field.

NRBOSDEP

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

If the NRBOSDEP 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 160 bytes long and should be filled with zeroes before first use.

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

To duplicate an NRD, wait until the initial CONNECT or OFFER has completed successfully, then copy the entire “working” NRB (up to and including the NRBOSDEP field) to a blank NRB at a different location. The second NRB can now be used for NetEx requests.

Appendix B. NRBSTAT Error Codes

Whenever a NetEx request is issued, the results of the request are returned in one or both of two fields, NRBSTAT and NRBIND. These 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 or not. 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 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 is returned as all zeroes. If a read-type command (SOFFER, DREAD) 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 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 thousands digit denotes the origin of the error; the low order three digits specifically identify tile error type. The codes for error origin are as follows:

0xxx	NetEx general. Errors detected by the user interface that prohibit proper processing of the command.
09xx	Reserved for implementation dependent errors in the user interface. In this implementation, these are socket errors from TCPIIP. Local names of error codes from TCP are defined in *>nsc>ntxr>include>nerror.h*
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 a remote host during the connection process.
9200-32767	Reserved.

0xxx and 90xx errors can be returned to any user program that accesses NetEx services. Normally, an application that accesses services at the session level only receives those errors (3xxx) related to session 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 is “rippled up” to the user rather than summarized at the higher level. For example, driver might report a “power off’’ or “not operational” 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 would be returned to the user along with a Disconnect Indication in NRBIND when the next user read command was issued.

Following that, the second digit was used to place the errors in rather loose categories:

x0xx	NetEx general or inconsistent NRB formats
x1xx	Specification errors in parameters passed to a particular protocol level
x2xx	Hardware errors
x3xx	Requests out of sequence and read timeouts
x4xx	NetEx initiated disconnect errors

x5xx Errors during connection

Note the following when using these codes:

- The error codes at each level have been made 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 someU1ing entirely different at the Session level.
- Some errors cause the loss of the connection or results in a connection not being established first. Any status code U1at 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 has a x100 (no N-ref) error returned to it.
- All errors that result in 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 NetEx request. A -1 means that request is still in progress.

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

H897IPV User Interface Error Codes

The following codes are H897IPV user interface error codes. Refer to the *NetEx Programmer's Reference Manual* for additional NRBSTAT error codes.

0000 Successful completion

0001 Pdata was truncated. A read-type operation completed normally, but the buffer provided by the user was not large enough to hold the data. NRBLEN and NRBUBIT reflect the amount of data received; however, the amount of data moved to the user's buffer was only the amount specified in NRBBUFL. The status of the connection is not affected.

0002 Invalid Pdata Buffer Address. NRBBUFL and NRBBUFA do not specify a block of storage that fits entirely within the user's addressable memory. The operation is suppressed. The status of the connection is not affected.

0004* Invalid Request. The request code (NRBREQ) is not valid. The operation is ignored and the status of the connection is not affected.

0005 Invalid Pdata Buffer Length. The buffer size specified (in NRBBUFL for reads and NRBLEN for writes) exceeds an implementation defined NetEx maximum. The operation is suppressed. The status of the connection is not affected.

0011 Odata was truncated. A read-type operation completed normally, but the Odata buffer provided by the user was not large enough to hold the data. NRBPROTL reflects the amount of data given to the user. The status of the connection is not affected.

0012 Invalid Odata buffer address. NRBPROTL and NRBPROTA do not specify a block of storage that fits entirely within the user's addressable memory. The operation is suppressed. The status of the connection is not affected.

0021 Odata and Pdata were truncated. A read-type operation completed normally, but both the Odata and the Pdata buffers were too small to hold the incoming data. NRBLEN and NRBUBIT reflect the

amount of Pdata received; however, the amount of Pdata moved to the user's buffer was only the amount specified in NRBBUFL. NRBPROTL reflects the amount of Odata given to the user. The status of the connection is not affected.

0100* No NREF specified. The user interface detected that the N-ref is not valid. The probable cause is the lack of, or a failing, CONNECT or OFFER.

0310 NRB for request is already in use. The user has attempted to re-use an NRB before a previous request with that NRB has completed. The 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.

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.

0900* (AAERROR) General interface library error. Contact your Network Executive Software Representative.

0901* (AAIO) I/O error, some physical I/O error occurred. Sometimes, this error may occur on a call following the one to which it actually applies.

0902* (AABADF) Bad file number.

0903* (AANOMEM) Not enough memory.

0904* (AAACCESS) Permission denied.

0905* (AAFAULT) Bad address.

0906* (AAINVAL) Invalid argument.

0907* (AANFILE) File table overflow.

0908* (AAIVIFILE) Too many open files.

0909* (AAWOULDBLOCK) Operation would block. An operation that would cause a process to block was attempted on an object in non-blocking mode.

0910* (AAINPROGRESS) Operation now in progress. An operation that takes a long time to complete (such as a connect()) was attempted on a non-blocking object.

0911 (AAALREADY) Operation already in progress. An operation was attempted on a non-blocking object that already had an operation in progress.

0912* (AANOTSOCK) Socket operation on non-socket.

0913* (A ADESTADDRREQ) Destination address required. A required address was omitted from an operation on a socket.

0914* (AAMSGSIZE) Message too long.

0915* (AAPROTOTYPE) Protocol wrong type for socket. A protocol was specified that does not support the semantics of the socket type requested.

0916* (AANOPROTOOPT) Option not supported by protocol.

0917* (AAPROTONOSUPPORT) Protocol not supported.

0918* (AASOKTNOSUPPORT) Socket type not supported.

0919* (AAOPNOTSUPP) Operation not supported on socket. For example, trying to accept a connection on a datagram socket.

0920* (AAPFNOSUPPORT) Protocol family not supported. The protocol family has not been configured into the system or no implementation for it exists.

0921* (AAAFNOSUPPORT) Address family not supported by protocol family.

0922* (AAADDRINUSE) Address already in use.

0923* (AAADDRNOTAVAIL) Cannot assign requested address.

0924* (AANETDOWN) Network is down. A socket operation encountered a dead network.

0925* (AANETUNREACH) Network is unreachable.

0926* (AANETRESET) Network dropped connection on reset. The host you were connected to crashed.

0927* (AACONNABORTED) Software caused connection abort A connection abort was caused internal to TCPIIP

0928* (AACONNRESET) Connection reset by peer.

0929* (AANOBUFS) No buffer space available.

0930* (AAISCONN) Socket is already connected.

0931* (AANOTCONN) Socket is not connected.

0932* (AASHUTDOWN) Cannot send after socket shutdown.

0933* (AATOOMANYREFS) Too many references! Cannot splice.

0934* (AATIMEDOUT) Connection timed out.

0935* (AACONNREFUSED) Connection refused.

0936* (AAHOSTDOWN) Host is down. A socket operation failed because the destination host was down.

0937* (AAHOSTUNREACH) Host is unreachable. A socket operation was attempted to an unreachable host.

0938 (TBADADDR) Incorrect address formal.

0939 (TBADOPT) Incorrect option formal.

0940 (TACCES) Incorrect permissions.

0941 (TBADF) Illegal transport fd.

0942 (TNOADDR) Could not allocate address.

0943 (TOUTSTATE) Out of state.

0944 (TBADSEQ) Bad call sequence number.

0945 (TSYSERR) System error.

0946 (TLOOK) Event requires attention.

0947 (TBADDATA) Illegal amount of data.

0948 (TBUFOVFLW) Buffer is not large enough.

0949 (TFLOW) Flow control.

0951 (TNODIS) The discon_ind not found on q.

0952 (TNOUDERR) The unitdata error not found.

0953 (TBADFLAG) Bad flags.

0954 (TNOREL) No ord rel found on q.

0955 (TNOTSUPPORT) Primitive not supported.

0956 (TSTATECHNG) State is in process of changing.

0959 (TNODATA) No data.

0980* (AANOCONN) No connection could be made. No server could be connected to because of problems with the interface library configuration file. More specific information should have been presented on standard error output.

0981* (AANOSERV) No server defined or selected. None of the available servers could be connected to. Check the status of the servers.

0982* (AABADHOST) Hostname could not be resolved. The IP address of the host named as server could not be discovered through standard means. Check the client host's IP configuration.

0983* (AASOCKBUF) Cannot set socket buffer size. The system call to set the socket buffer sizes failed.

0984* (AARECVZER) Zero bytes received on socket *recv*. The TCP connection was broken by the server in an unexpected manner. Check the status of the server.

0985* (AABADINTR) Cannot convert the interface address.

0990* (AASOCK) Error on socket creation. Contact your Network Executive Software representative.

0991* (AACONN) Error on connect. Contact your Network Executive Software representative.

0992 (AASEND) Error on send. Contact your Network Executive Software representative.

0993* (AARECV) Error on receive. Contact your Network Executive Software representative.

0994 (AACLOS) Error on close. Contact your Network Executive Software representative.

0995* (AADISC) Error on disconnect. Contact your Network Executive Software representative.

1101 DWRITE invalid datamode. The datamode specified for this DWRITE request is invalid. The request is rejected. The status of the connection is not affected.

1103 DWRITE invalid Odata length. The Odata length (NRBPROTL) specified for this DWRITE request is invalid. The length of the message proper (NRBPROTL) must be between 8 and 64 bytes inclusive. The request is rejected. The status of the connection is not affected.

1300 DRIVER LEVEL - READ TIME OUT. A DREAD request timed out before any data was received for this connection. The time value used for the timeout was in NRBTIME. No data was received. The status of the connection is not affected.

1310 DRIVER LEVEL - READ DATA TIME OUT. Data that was received for this connection has been discarded because the application did not issue a read request for a period of time greater than the data timeout period (30 seconds). The status of the connection is not affected.

3300 SESSION LEVEL - READ TIME OUT. A session level request (SREAD or SOFFER) timed out before any data was received for this session. The time value used for the timeout was in NRBTIME. No data was received. The status of the connection is not affected.

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 continues to wait for the confirm or disconnect request.

3303 An SCONNECT request was previously issued. The only requests allowed after the SCONNECT are SDISCONNECT to disconnect, or SREAD to read the Confirm or Disconnect indication. The request is rejected. NetEx continues to wait for the SREAD or SDISCONNECT request.

3310 SESSION LEVEL- READ DATA TIME OUT. Data that was received for this session has been discarded because the application did not issue a read request for a period of time greater than the data timeout period (30 seconds). The session is terminated.

7171 Application failed to give control to the interface library often enough to allow “keepalive requests” from the TNP to be able to respond.