H267IP/H267IPI NetEx/IP® Requester

for OpenVMS on HP Integrity Systems

Release 7.0

Software Reference Manual

MAN-REF-H267IP-03

Revision Record

Revision	Description
01 (07/91)	Manual released by NSC. This manual corresponds to release 1.0.
MAN-REF-H267IP-01 (01/2001)	Manual updated corresponding to version 5.0. A glossary was added to explain terms in this manual. Manual part number changed to reflect new copyright ownership. Revision number reset to 01 to reflect release of new NETEX/IP product.
MAN-REF-H267IP-02 (01/2002)	Manual updated for version 6.0. Documentation added for NESi Software Licensing and NetEx Protocol 4 facilities. World wide web and e-mail addresses updated for NETEX.COM domain.
MAN-REF-H267IP-03 (x/2009)	Manual updated for version 7.0 which added support for OpenVMS Integrity systems. Other minor documentation changes and updates.

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

Comments may also be submitted over the Internet by addressing email to:

`pubs@netex.com`

or, by visiting our web site at:

www.netex.com

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

Preface

This manual describes the H267IP and H267IPI NetEx/IP® Requester interface software for the HP OpenVMS Alpha and OpenVMS Integrity systems.

“Introduction”, “NETEX Session Services”, and “NETEX Request Block” are intended for all readers. “User Interface Library” describes the library of subroutines that are called by the C high level language program.

“Installation” and “Installation Procedure” describe NetEx/IP installation and are intended for the systems programmer installing NetEx/IP. The Network Executive Software *NESiGate NetEx/IP Offload LAN-to-IP Gateway Reference Manual* describe in detail the NetEx/IP operator interface.

Appendices include a list and descriptions of the error messages and codes issued by NetEx/IP, Configuration Manager messages, NCT Loader error messages, and asynchronous post-exit processing routines.

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**
460364	*H261 Bulk File Transfer (BFX™) Utility for OpenVMS Operating System Software Reference Manual*
460376	*H263 USER-Access® for OpenVMS User Guide*
460757	*"C" Configuration Manager and NETEX Alternate Path Retry (APR) User Guide*
460280	*NCT Loader Software Reference Manual*
460580	*NETEX Application Programmer's Interface Software Reference Manual*
MAN-REF-LOSW-01	*NESiGate NetEx/IP Offload LAN-to-IP Gateway 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**
Hewlett Packard Computers, Inc.	**HP, OpenVMS, Integrity, Alpha**

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 to be used 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: <uparrow>).
<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 NESiGate LAN Offload adapter 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).

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 which 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 Configuration Table Loader (NCTL): An interactive NETEX application program used for configuring NESiGate LAN Offload adapter, updating their NETEX configuration parameters and/or Network Control Table. The NCT Loader takes a PAM file created by the Configuration Manager and transfers it to the NESiGate LAN Offload adapter through a NETEX connection.

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 in the NESiGate LAN Offload adapter. The latter case use 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, et cetera) 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-networked 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.

Table of Contents

Figures

Introduction

NETEX 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, such as this version for use with the HP OpenVMS operating system. All of these versions provide a common high-level interface to simplify programming requirements. NETEX utility programs are also available, such as the Bulk File Transfer (BFX™), Print File Transfer (PFX™), and USER-Access® utilities.

H267IP/H267IPI

In a NetEx/IP requester configuration, the NetEx software resides and executes on the NESiGate LAN Offload adapter. H267IP/H267IPI resides on the HP host and is the user interface to the NESiGate LAN Offload software. H267IP/H267IPI passes NETEX requests and data to and from the NESiGate adapter by way of the host’s Ethernet using TCP/IP.

Figure 1 shows an example of H267IP/H267IPI running in a NESiGate NETEX/IP environment.

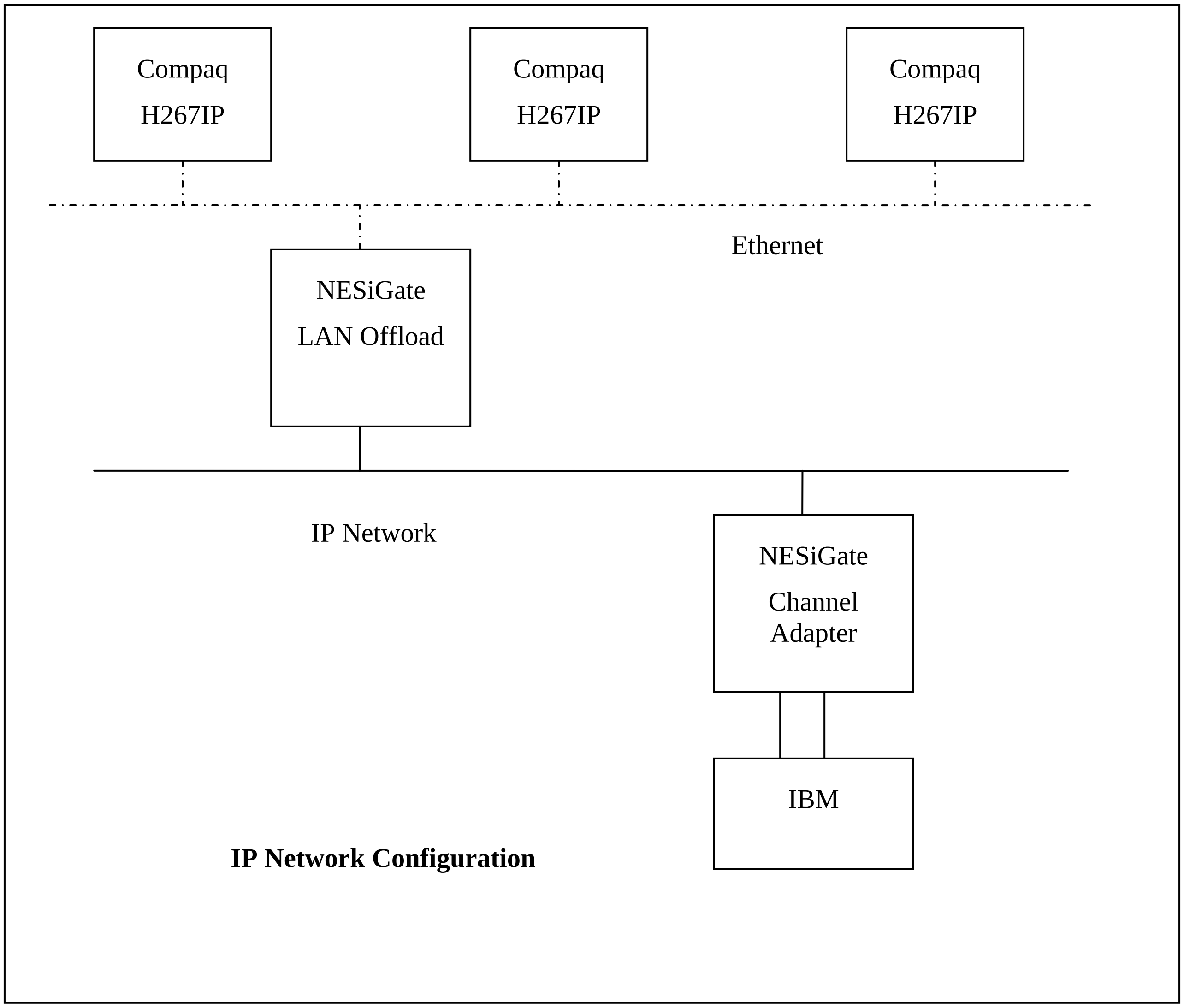

Figure 1. Network Configuration Examples

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

New Features

License Keys

H267IP/H267IPI will not function unless a valid license key is obtained from Network Executive Software's Customer Support and placed in a configured keys file. See the installation section of this manual for further details.

Protocol 4

This is a new feature of the NESiGate Offload NetEx. A new transport protocol has been implemented which provides higher bandwidth utilization over long, high-latency paths, such as satellite connections. It is invoked when both the local and remote hosts have "PROTOCOL=4" specified in the NCT. A host may support both protocol 4 and protocol 2.

NETEX Characteristics

The following sections describe the characteristics of the NETEX software.

- External interface
- Internal interaction
- NETEX connections
- Design flow efficiency and flexibility
- User exits
- 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 C or other high-level languages. 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 are consistent and allow 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 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 Executive Software device drivers.

Once NETEX accepts data from the caller, NETEX must deliver 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 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 multi-megabit 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.

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 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 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. See "REMOTE Command" for more information. Security features are provided.

Basic I/O Flow

Figure 2 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 available network hardware to communicate with the calling program on the other processor.

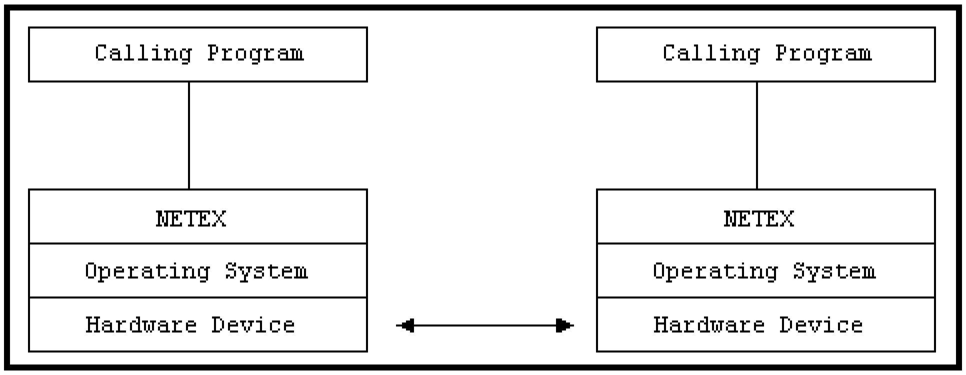

Figure 2. Basic I/O Flow

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. 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.

NESiGate Offload NETEX

The NETEX program resides as either LAN Offload or Channel Offload software running in the NESiGate adapter. Only the Hxx7IP NETEX Requester user interface program resides on the host. In this implementation, H267IP/H267IPI and the HP OpenVMS TCP/IP products are used for transport of NETEX requests and buffers between H267IP/H267IPI, the host, and the NESiGate adapter.

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.

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-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

Although each layer physically interacts only with adjacent layers, each layer appears to communicate directly with the corresponding layer of the other model. Figure 3 illustrates this concept.

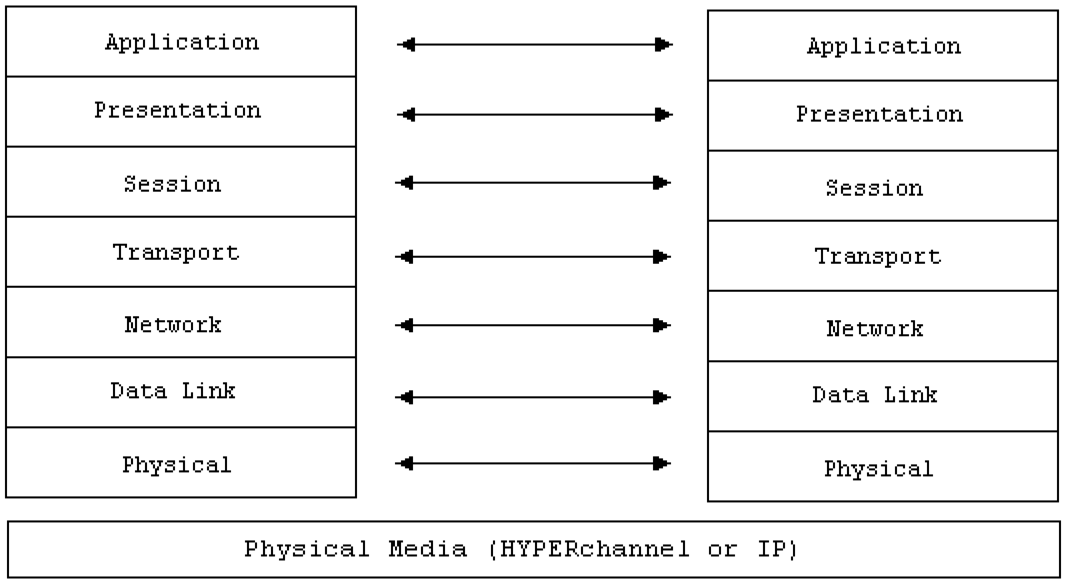

Figure 3. ISO Model Communication

> **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.

In the NetEx/IP requester and NESiGate Offload configuration, the NetEx/IP protocol stack is executed in the NESiGate Offload Adapter. NetEx/IP requester running in the host machine consists of NetEx/IP protocol library “calls” which package NetEx/IP commands and data and transmit them to NESiGate adapter. The NetEx/IP protocol stack interfaces to the IP network via UDP/IP and NESiGate’s physical IP media. Figure 4 shows that the NESiGate adapter’s network interface hardware and firmware form the lower two layers.

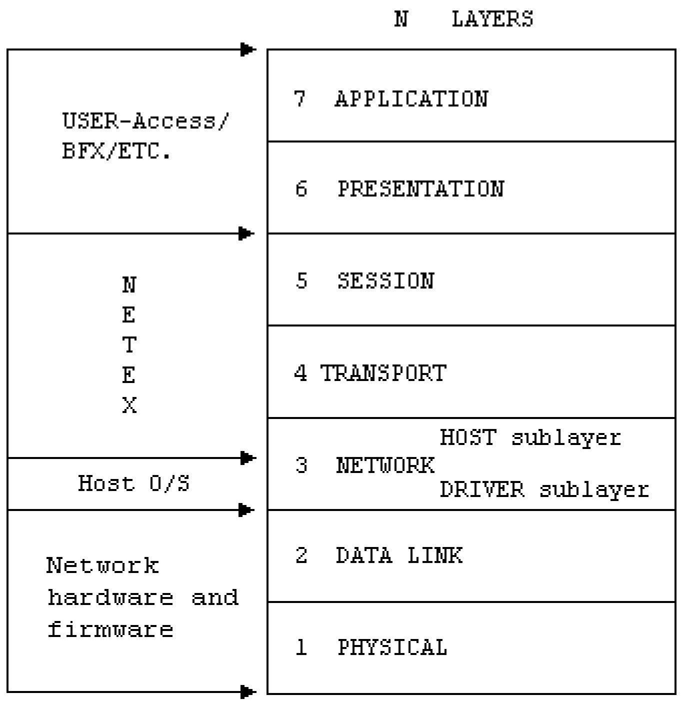

Figure 4. NETEX and the ISO Model

Session Layer Services

As the highest layer within NETEX (referring to the ISO model in Figure 4), the NETEX 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 described in "NETEX 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. 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-to-NETEX 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 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 HYPERchannel 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 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 the adapter type and requested by the user's data mode parameter.

Installation

The installation procedures documented in the software reference manual for this product are no longer valid. Please follow the installation instructions in the H267IPI “Memo To Users” document.

NETEX Session Services

As discussed in “Introduction”, the user may interact with NETEX at the session and driver layers. This section describes interaction at the session layer, since most users will be programming at that level. Users wishing to program at lower levels should still study this section to understand the basic concepts and programming techniques that apply to using NETEX. Users programming at the session layer do not need to study any of the other interfaces, since all layers below are fully managed by the session layer software.

To communicate using the session layer of NETEX, the calling programs (that is, the programs that are calling NETEX) 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. The NETEX transport is an important feature of NETEX, it uses internal error checking and error recovery procedures. Once NETEX 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 section explains the concepts of session layer “intertask” communication using NETEX. 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 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. The NRB is updated by the calling program when the program issues requests (either directly or by way of NETEX), and it is updated by NETEX when requests are completed by NETEX. The NRB is completely described in “NETEX Request Block”.

The NETEX session requests, listed in the approximate order which they are issued, are:

SOFFER	This command makes a calling program using NETEX available to another program on either a remote host or the 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 the amount of data sent does not exceed the maximum segment size allowed on either the sending or receiving host.

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 (data block size, etcetera) specified in the SCONNECT are accepted. This request may also write data to the connecting program, provided the amount of data sent does not exceed the maximum segment size allowed on either the sending or receiving host.
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 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 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 completes the specified request(s). For SWRITE, SCLOSE, SCONNECT, SCONFIRM, or SDISCONNECT requests, NETEX 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 terminate a connection. It may contain data 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 a 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 sections.

General Concept of a Session

Before explaining in detail how each part of a session is programmed, the following sections explain the general concept of a simple NETEX session.

Figure 5 shows a simplified example of a session where one program reads data from another.

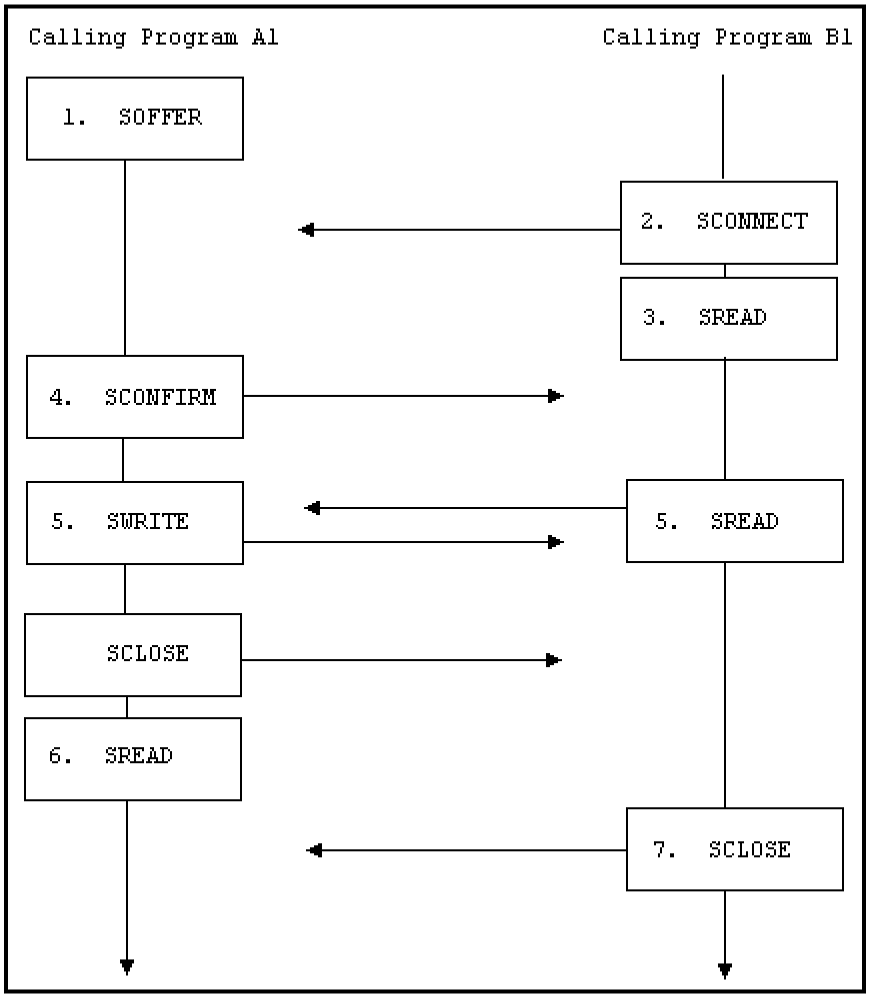

Figure 5. General Concept of a Session

The following text describes the session flow shown in Figure 5.

1. Calling program A1 in host A issues an SOFFER. This indicates that program A1 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 (that is, when it is accepted by NETEX), program B1 issues an SREAD to prepare to receive program A1's response to the SCONNECT.
4. When the SCONNECT is received by the NETEX 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 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 indicates 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 (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.

This figure references an NRB. An NRB (discussed in “NETEX Request Block”) is a block of parameters used to signal requests to NETEX and to return status to programs.

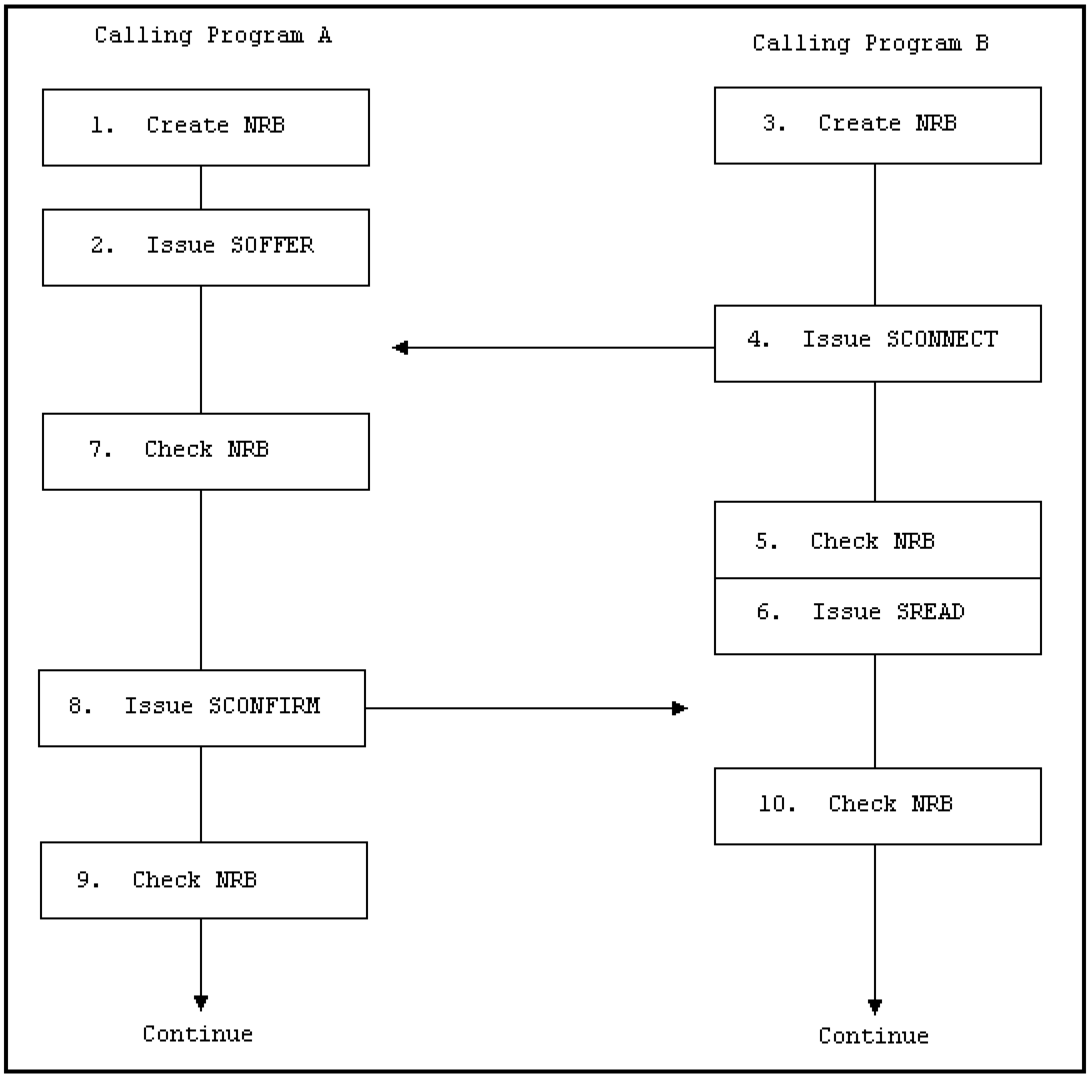

Figure 6. Establishing a Connection

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 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 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 data such as a password
5. Program B then checks the NRB to determine the status of the request. The NRB indicates one of the following:
   - a request is in process
   - a request has completed successfully
   - the request has generated an error

Figure 6 continues assuming the NRB indicated normal completion. If the NRB indicates that a request is in process, 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 a 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 will 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 a 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. If it was, program A will continue with the session. If NETEX did not accept the SCONFIRM, program A will 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 a 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 session. It also introduces the concept of using the NRB for communication with NETEX. After requests are issued, the NRB is examined to see when NETEX completes the request. This may be done using the SWAIT request, or by periodically checking the NRB fields. The NRB fields are discussed in "NETEX Request Block".

Data Transfer

Unlike the rules for establishing a session, NETEX provides a lot of flexibility in how session requests are used for the data transfer process. The following sections 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. Usually, 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.

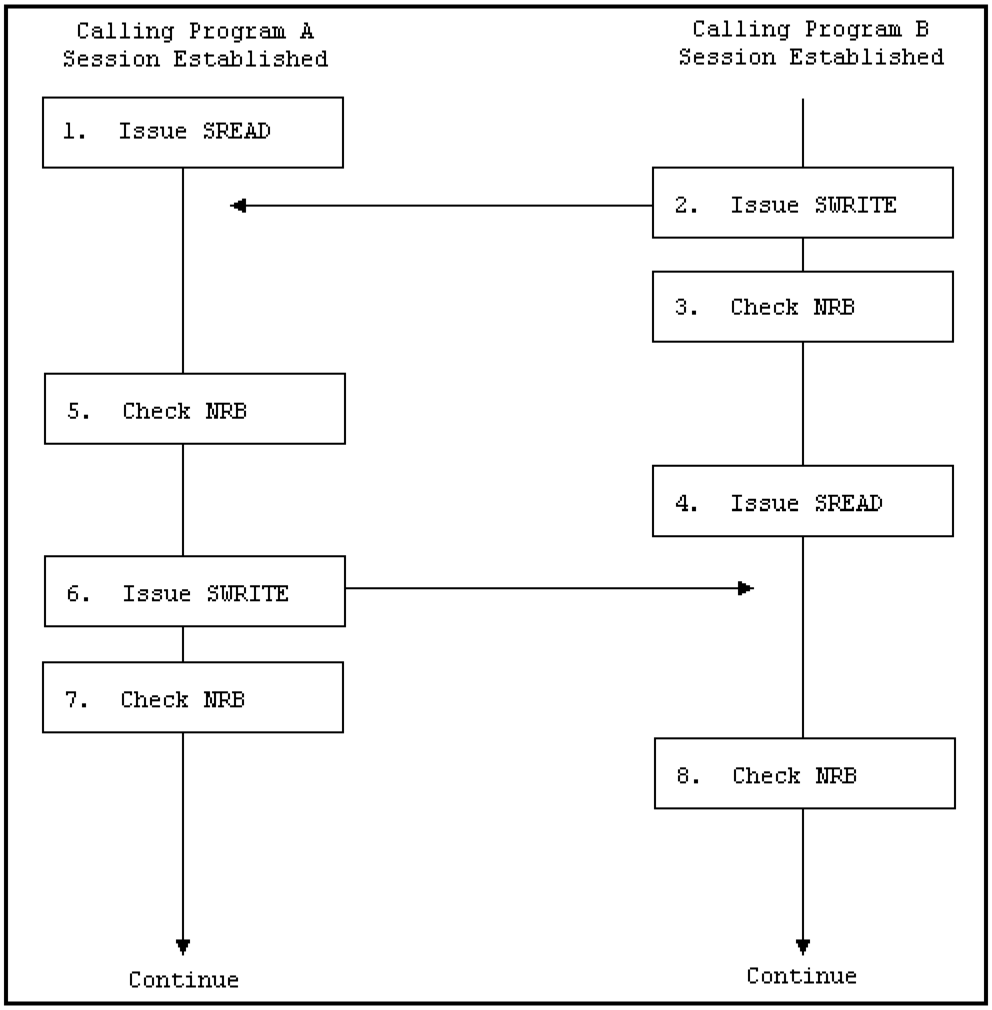

Figure 7. Write/Read Data Transfer

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 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 accepted the SWRITE or indicated an error.

   Once NETEX has accepted the data, NETEX 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 will act appropriately.

   If program B issued an SDISCONNECT, program A will 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 accepted the SWRITE by checking the NRB. If the NRB indicates the SWRITE was not accepted, program A will 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 “Abnormal Session Termination” .

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 used for satellite communication, discussed in “Satellite Communication”, but it is also well-suited for local data transfer.

Because NETEX 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 establishes 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 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 acknowledgment or another message.

Note: 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 in “Terminating a Session”.

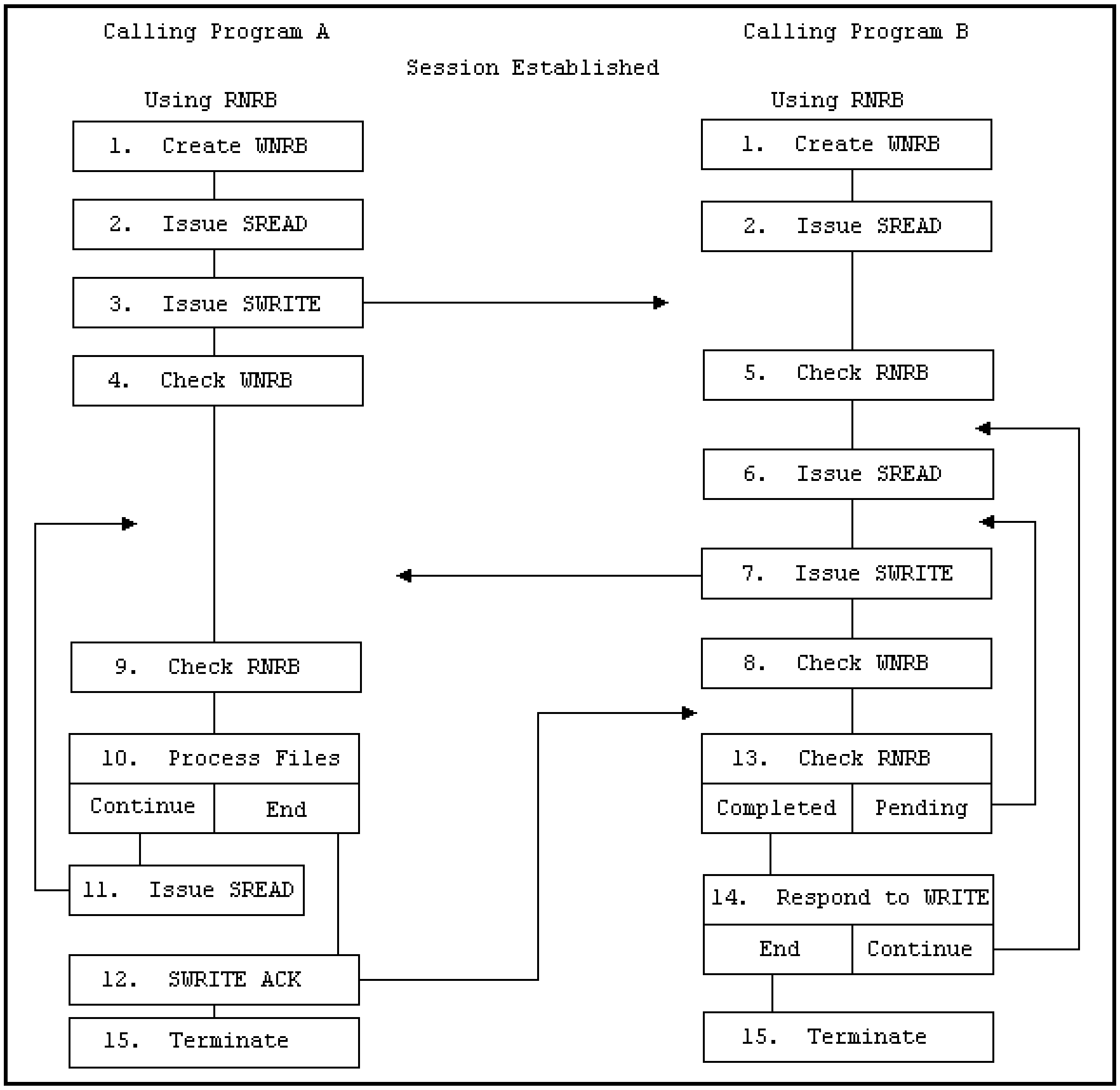

Figure 8. Concurrent SREAD and SWRITE requests

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 events 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 which will be used with the SWRITE requests
2. Both programs issue an SREAD request to prepare to receive requests from the other program. The RNRB is specified in the SREADs as the place for NETEX 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 to respond to that request.
4. Program A checks the WNRB to see if NETEX 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, receives 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 which 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 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 wishes 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 acknowledgment 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, receiving an acknowledgment, recording a message, terminating the session, etcetera.
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. Session termination is described in “Terminating a Session”.

The previous example demonstrates the technique of using SREAD and SWRITE requests concurrently. Waits should only be used after SWRITE requests when using this technique. Abnormal terminations are discussed in “Abnormal Session Termination”.

One-Way Data Transfer

A typical use of NETEX 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).

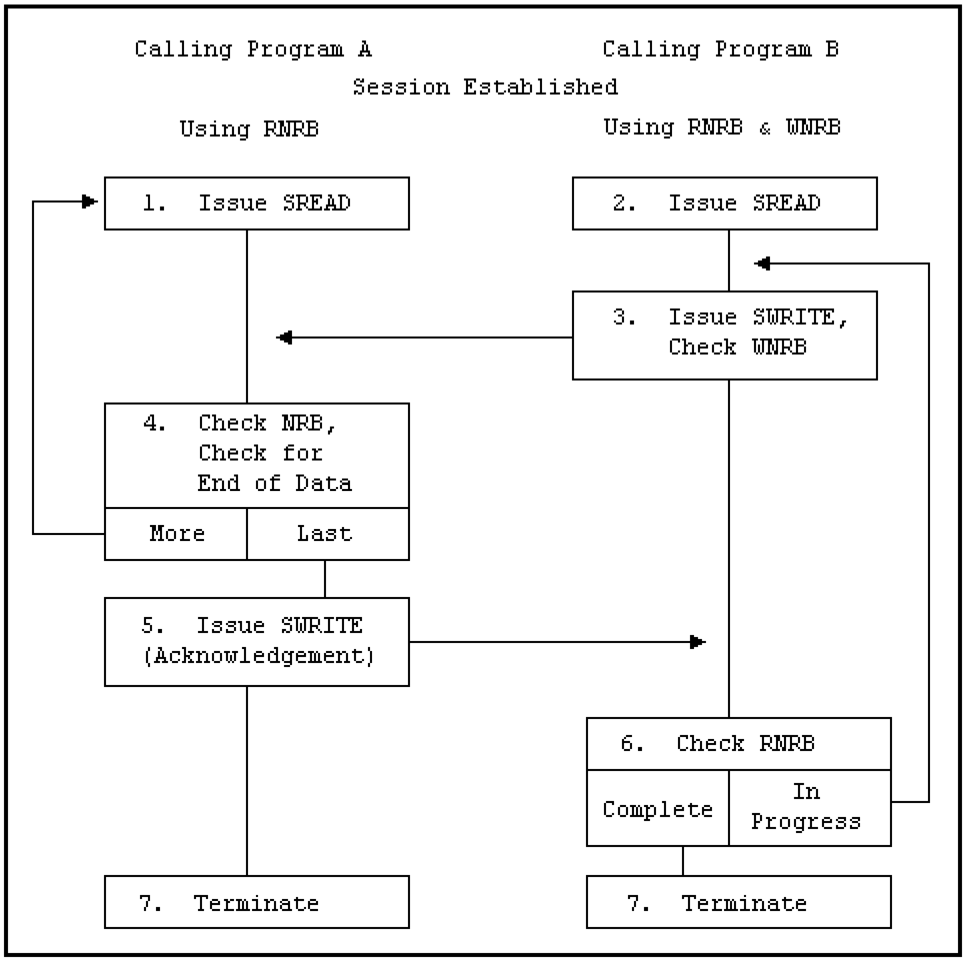

Figure 9. One-Way Data Transfer

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 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 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 has verified the integrity of the data.)
6. Program B checks the RNRB. If RNRB 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 is-

sue an SWAIT to suspend execution until program A returns a response. When the RNRB completes, program B checks the data written by program A. Normally, this would be an acknowledgment 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. Terminating a session is described in the following section.

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.

Abnormal terminations are discussed in the following section.

Terminating a Session

NETEX 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 which the events occur.

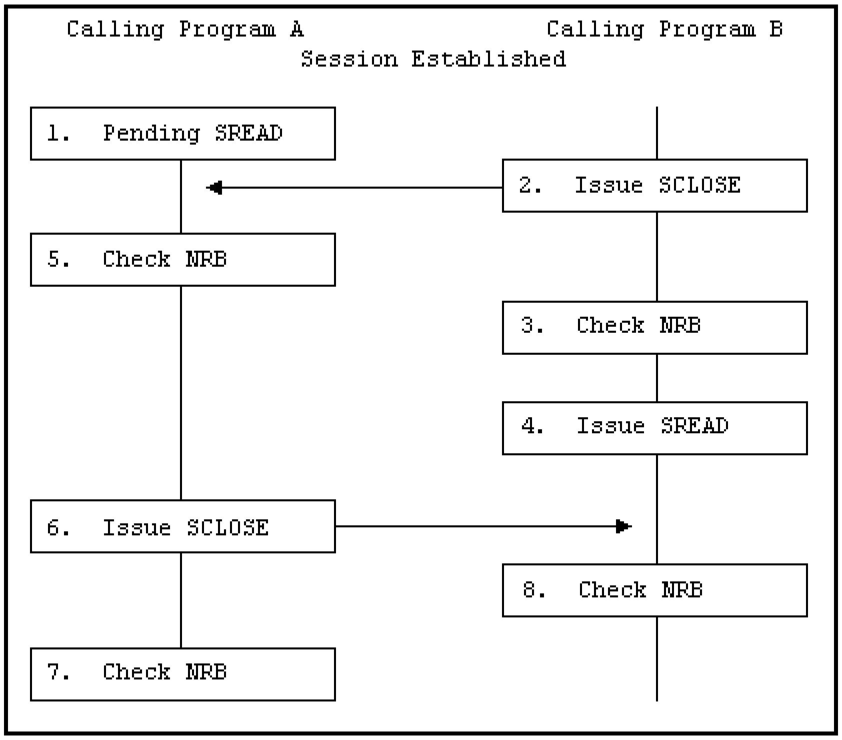

Figure 10. Normal Session termination

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. If it was accepted, program B may close output files or perform other termination processing. No other NETEX 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 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.

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.

Sessions may be unexpectedly terminated by the other program for various reasons depending on how the program is written. Some typical reasons for immediate termination 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 eliminated by reconnecting with the calling program.

NETEX 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 simpler. The following topics are discussed:

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

Handling Multiple Connections

NETEX 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 requestor 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 (a requestor) establishes a connection, the server SREADs or SWRITEs files to or from its database as specified by the requestor.

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 though 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 requestors at one time.

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

Service WAIT Options

On each session call, the user has the option to wait or not to wait for the request to complete. If waiting is desired, then the "W" form of the call should be used. The User Request Manager will issue the wait on the user's 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 zero 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 and a real-time application, use SWAIT(0) to wait on zero NRBs, then check the real-time device. When issuing an SWAIT on zero NRBs, check the NRBSTAT fields of the NRBs for the requests that you are interested in.

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.

Satellite Communication

The standard NETEX requests may be used when satellite communications facilities are a part of the network. NETEX 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 inherent in the satellite subsystem (approximately 600 microseconds), you must keep the communications channel "full" of data. Do this by using concurrent SREADs and SWRITEs which transmit data in large amounts before having the writing application stop to wait for an acknowledgment 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 acknowledgment written in the other direction, the propagation delay would have major impact. But, if an entire file is transmitted before an acknowledgment 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 link, NETEX, or the other host, there is no way to know how much unacknowledged data was successfully received.

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

NETEX Error Recovery Procedures

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

Error Codes

Whenever a NETEX 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 simpler matter.

NRBSTAT indicates whether an operation is in progress. If the operation is no longer in progress, NRBSTAT also indicates whether the operation 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 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, then 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 205.

Common Error Recovery Procedures

The following are commonly encountered errors with an explanation of how to recover from them:

- Other program not there - Operators or users must coordinate the running of the two NETEX programs so that one has not timed-out before the other has had a chance to establish a session.
- Other program busy - Retry NETEX after a suitable delay.
- NETEX 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 provides for common types of code conversion by using NESiGate LAN Offload adapter hardware or NETEX software facilities. The calling program uses the datamode (NRBDMODE) word of the NRB to specify either manual or automatic code conversion.

If manual conversion is selected, the caller completely specifies which assembly/disassembly and code conversion functions will be used on both adapter output and adapter input. The caller must determine which assembly/disassembly modes and code conversion tables are significant to the adapters.

Note: Manual mode is not supported on this version of NETEX.

If automatic code conversion is selected, the caller simply specifies the source character set and the destination character set. NETEX then uses any code conversion hardware that is available, and carries out other code conversion using software when necessary.

NETEX Request Blocks

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 by way of 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 will be connected to more than one program at a time, several NRBs must be used.

The use of the NRB fields vary slightly between the different levels of programmer interface. Specific differences are described in the individual field descriptions that follow.

Rules for NRB Use

The following principles are designed to make high level language use of NETEX somewhat transportable between machines.

- Before initiating a connection, the user must clear the NRB, including the operating system 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 all 40 fields 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 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, Tandem is 8 bits, and so on.

NRB Fields

Figure 11 shows the fields in the NRB. All NRB fields are two words long, (32 bits). The NRB contains forty fields.

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. Subsequent attempts to change these fields will have no effect.

NRB fields may be referenced by programs using the names shown in Figure 11 if the Structure definitions shown in Figure 12 are used. Otherwise, the NRB fields must be referenced using the index numbers shown on the left side of Figure 11.

0	NRBSTAT	NETEX request status returned to user
1	NRBIND	Data type indication returned to OFFER/READ
2	NRBLEN	Length of data
3	NRBUBIT	Unused bit count
4	NRBREQ	User request code
5	NRBNREF	NETEX reference number identifying the connection
6	NRBBUFA	Starting address of user's data 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 READ requests
13	BRBBLKO	Maximum buffer size for WRITE requests
14	NRBPROTA	Address of user's Odata buffer
15	NRBPROTL	Length of Odata
16	NRBRESV1	Reserved
17	NRBRESV2	Reserved
18 19	NRBOFFER/ NRBPAM	Session Level – offer name Transport/Network Level – Pointer to PAM
20 21	NRBHOST/ NRBRREF	Session Level – Remote host name Transport/Network – Remote reference number
22	NRBRESV3	Reserved
23	NRBRESV4	Reserved
24 to 39	NRBOSD	Operating system dependent data

Figure 11. NETEX Request Block (NRB) Fields

```
                   STRUCT     NRB DEF (*)
                      BEGIN
                      INT(32)    NRBSTAT;
                      INT(32)    NRBIND;
                      INT(32)    NRBLEN;
                      INT(32)    NRBUBIT;
                      INT(32)    NRBREQ;
                      INT(32)    NRBNREF;
                      INT(32)    NRBBUFA;
                      INT(32)    NRBBUFL;
                      INT(32)    NRBDMODE;
                      INT(32)    NRBTIME;
                      INT(32)    NRBCLASS;
                      INT(32)    NRBMAXRT;
                      INT(32)    NRBBLKI;
                      INT(32)    NRBBLKO;
                      INT(32)    NRBPROTA;
                      INT(32)    NRBPROTL;
                      INT(32)    NRBRESV1;
                      INT(32)    NRBRESV2;
                      STRING     NRBOFFER[0:7];
                      INT(32)    NRBPAM = NRBOFFER;
                      STRING     NRBHOST[0:7];
                      INT(32)    NRBRREF = NRBHOST;
                      INT(32)    NRBRESV3;
                      INT(32)    NRBRESV4;
                      INT(32)    NRBUSER = NRBESV4;
                      INT(32)    NRBOSDEP[0:15];
                      END;
```

Figure 12. NRB Structure Definitions

The following sections describe the fields in the NRB shown in Figure 12.

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 is explained in Appendix A.

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 non-zero value.

The values returned in NRBIND are defined as follows:

NRBCNIND (1)	Connect Indication
NRBCFIND (2)	Confirm Indication
NRBDTIND (3)	Normal data Indication
NRBEXIND (4)	Reserved
NRBCLIND (5)	Close indication
NRBDCIND (6)	Disconnect indication
NRBSTIND (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, non-zero value. If NRBSTAT is non-zero, 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 wants to send exactly 35 of its 60-bit CM words to an IBM processor and wants it returned at a later date. The CYBER user will specify NRBLEN=35 and NRBUBIT=0. Datamode will be bit stream. NETEX will record that 35*60=2100 bits of information was sent over the network. The IBM user will 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 wants to send 151 ASCII characters (8*151=1208 bits) to a Unisys and have them converted to Field-data. The CRAY user can specify NRBLEN=19 (64 bit words) (64*19=1216 bits) and NRBUBIT=8 (bits) since seven characters will be in the trailing CRAY word. The CRAY driver will send the ASCII over the network and record that 8*151=1208 bits of information were sent. The Unisys will select sixth-word A/D mode and code conversion and determine that (1208/8)*6=906 bits of information will result. It will 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: Those programs that do not need the NRBUBIT can ignore its existence, knowing that 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 will 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

The option flags refer 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

The service level indicates whether 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:

xx01	Connect (valid for S, T, and N levels)
xx02	Confirm (valid for S, T, and N levels)
xx03	Write (valid for S, T, and N levels)
xx04	Reserved
xx05	Close (valid for S and T levels)
xx06	Disconnect (valid for S, T, and N 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 a 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.

There are two forms of datamode: manual and automatic.

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 appropriate 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 A/D 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 H267IP/H267IPI DEC OpenVMS version of NETEX.

Manual datamode has the following format:

‘1’ The manual mode indicator “1” is in the high order bit.

Outgoing A/D

These bits identify 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

These bits identify 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’ A second manual mode indicator “0” is in the high bit of the low order byte.

Incoming A/D

These bits identify 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

These bits identify 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.

Automatic Datamode

Automatic datamode is designed for all common NETEX transfers. When automatic 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.

Automatic datamode supports three conversion options:

Bit Stream

The bit pattern sent is precisely reproduced in the destination machine.

Octet

Eight-bit binary quantities are sent from one machine to another, using an 8-bit byte representation appropriate to each machine.

Character

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 automatic datamode has the following format in the NRBDMODE field:

’0’ The automatic datamode indicator “0” is in the high order bit.

Source Character Set

These bits identify the conversion option (from 2) of the data used in the write buffer of the transmitter.

’0’ The high bit of the low order byte is reserved.

Destination Character Set

These bits identify the conversion option (from 2) 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.

Table 2. Auto Datamode Character Sets

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

The processing rules for automatic datamode are listed below:

- The transmitting driver examines the source character set specified. The character set implicitly specifies the method used to represent those characters on the transmitting machine. The driver selects an A/D mode so those characters will be 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 the 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 A/D mode to convert the 8-bit quantities coming over the 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 to 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 Network Configuration Table (NCT) (see "Installation").
1	Use Version 1 NETEX protocol. This protocol is used in Release 1 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 the 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.

Note: 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 non-zero 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. 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 the following values controlling these buffer sizes:

1. The default input and output block sizes to be used, if these are not specified (left zero) by the caller.
2. 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.

Two default options are 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. 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.

Note: 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 will be used to send and receive data in this connection. These limits are dependent on the following:

- The buffer capacities generated in both the local and remote copies of NETEX.
- 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, the Network

Service will return the maximum that is available if the caller's size request is not available. The caller must scale its 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," which 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, and so on).
- 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. NESiGate LAN Offload utilities do not use Odata. Consult Network Executive Software personnel before using Odata to determine whether it is available.

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.

NRBPROTL always contains the length of the Odata in octets, not "addressable units."

The protocol field has special significance when used with 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 and NRBPAM

The use of this field varies depending on the layer it is issued in. If this field is used in the Session layer, it is called NRBOFFER. If this field is used in the Transport or Network layer, it is called NRBPAM.

NRBOFFER for Session Requests

Used with a session level request, 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 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 significant in ASCII should be used during the name matching process.

NRBPAM for Transport and Network Requests

Used with transport or network level requests, NRBPAM specifies the address of the buffer that will hold the incoming PAM. This PAM is a complete description of a network path that will allow communications to take place between both parties. This PAM may be examined by the offering application to determine the identity of the party that is contacting it. The size of this buffer should be 128 octets.

NRBHOST and NRBRREF

The use of this field varies depending on the layer it is issued in. If this field is used in the session layer, it is called NRBHOST. If this field is used in the transport or network layer, it is called NRBRREF.

NRBHOST for Session Requests

Used with a session level request, NRBHOST specifies the symbolic name of the host computer that will 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 eight characters in length. Names less than eight characters long should be padded with blanks.

NRBRREF for Transport and Network Requests

Used with a transport or network level request, NRBRREF contains the Nref used by the remote party in the NRBNREF field. If a Connect Indication specifying the proper local Nref arrives which does not contain the proper remote Nref, then the incoming message will be ignored. If a zero is specified in NRBRREF, then any remote Nref will be acceptable. In that case, NRBRREF will contain the remote Nref when the offer completes.

NRBRESV3

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

NRBUSER

This field is reserved for users. Usually, it will be used to pass information to the User Exits that exist in Host based NETEX implementations. NETEX will not process this field in any way.

NRBOSD

NRBOSD is reserved for internal use. NETEX software uses this field to service and monitor the progress of NRB requests. The contents of these fields is maintained by NETEX 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 sections.

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 requests.

C High Level Interface

NETEX includes a library of subroutines that are designed to be called by C high level language programs. Also included are global data declarations and external reference declarations that may be included at compile time. This file is called *netex.h*. 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.

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	disconnect (immediate)
netxchek	check completed files
sparam	change DXCILRC file or TCPIP process
spname	assign server name

These calls are described in "NETEX Session Services". The formats of the calls are presented using the conventions stated in the *NetEx/IP Offload LAN-to-IP Gateway Reference Manual*.

C NETEX Request Blocks

The C user creates an NRB by declaring a data structure of 21 fields using the NETEX supplied data structure template NRB. Various fields of this record will hold the information to be transferred to NETEX, and others will contain the information that is returned by NETEX. NRB variables should be declared to be of that type. Before these NRB records are used for any NETEX request, Network Executive Software advises to zero all the elements of the record. This will allow defaults for fields not explicitly used by the caller to take effect.

The NETEX C subroutines have the philosophy that arguments commonly passed to NETEX (such as a data buffer address) will be passed as parameters to the subroutine. "Exotic" parameters to be passed to NETEX, such as maximum input block size, will be supplied by storing the desired value in the proper field of the NRB record. When the request completes, the C program directly accesses the desired fields of the NRB record to determine whether the operation completed properly. If NRB is declared as a 21 field record, the fields of the record will be defined using the following:

Field	Type	Function
NRBSTAT	long	Status returned from NETEX
NRBIND	long	Data type indication from OFFER/READ
NRBLEN	long	Length of data received in words
NRBUBIT	long	Unused bit count
NRBREQ	long	Request code
NRBNREF	char*	NETEX Reference number (N-ref)for this session
NRBBUFA	long	User’s Pdata buffer address
NRBBUFL	long	Length of the buffer
NRBDMODE	long	Datamode for WRITE request
NRBTIME	long	Timeout for a read type request (in seconds)
NRBCLASS	long	Class of service
NRBMAXRT	long	Maximum rate of data transmission
NRBBLKI	long	Blocksize to use for input
NRBBLKO	long	Block size to use for output
NRBPROTA	char*	User’s Odata buffer address
NRBPROTL	long	Length of Odata buffer
NRBRESV1	long	Reserved
NRBRESV2	long	Reserved
NRBOFFER	char:8	(Session Level) Offer Name
NRBPAM	long	(Source) (Network & Transport Level) Pointer to PAM (nrbpam=nrboffer)
NRBHOST	char:8	(Session Level) Logical name of destination host
NRBRREF	long	(Network & Transport Level) Remote Reference Number (nrbrref=nrbhost)
NRBRESV3	long	Reserved
NRBRESV4	long	Reserved
NRBUSER	long:1	Free for user to assign (nrbuser=nrbresv4)
NRBOSDEP	long:16	Reserved for system dependent information

Figure 13. 'C' NETEX Request Block (NRB)

SOFFR C Function

The SOFFR (offer) and SOFFRW (offer wait) functions make the services, provided by the calling NETEX application program, available to programs running on other hosts. The SOFFR is actually a specialized form of read request. The SOFFR reads any data associated with the SCONN.

Before issuing an SOFFR call, the user must provide an NRB (described in "NETEX Request Block") to be used by the user interface.

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 were shown in the SOFFR function format. The parameters must be specified in the order presented. If parameters are omitted, then the commas must be used to preserve subsequent parameters' positions.

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

(struct nrb *) This required parameter is a pointer to the NRB data area to be passed to NETEX.

buffer

(extptr char *) This required parameter is the extended address of the buffer data area to receive data sent by the corresponding SCONNECT request.

length

(long:VALUE) 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 field will contain the number of bytes of information actually sent to the offering application.

timeout

(long:VALUE) 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

(char*:REF:) 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 must be provided as a string in the CALL statement padded with blanks to eight characters in length.

SOFFR 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 S-conn 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.

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 were shown in the SCONN function format. The parameters must be specified in the order presented. If parameters are omitted, then the commas must be used to preserve subsequent parameters' positions.

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

(struct nrb*) This required parameter is the pointer to the NRB data area to be passed to NETEX.

buffer

(extptr char*) This required parameter is the extended address of the buffer data area that holds the user data to be sent to the corresponding application.

length

(INT(32):VALUE) This required parameter is the length of the data (in addressable units) to be sent to the corresponding program. This value is 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.

datamd

(long:VALUE) This required parameter is the datamode to be used to send the connect data to the corresponding application. Refer to NRBDMODE in "NETEX Request Block" for a discussion of the datamode parameter.

pname

(char*REF:) 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 must be provided as a string in the call invocation, padded with blanks to eight characters in length.

hname

(char*REF:) 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 must be provided as a string in the call invocation, 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
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 be optionally written during the confirm process with this command.

Before issuing the SCONF function, 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 were shown in the SCONF function format. The parameters must be specified in the order presented. If parameters are omitted, then the commas must be used to preserve subsequent parameters' positions.

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

(struct nrb*) This required parameter is a pointer to the NRB data area to be passed to NETEX.

buffer

(extptr char*) This required parameter is the extended address of the buffer data area that holds the user data to be sent to the corresponding application.

length

(long:VALUE) This required parameter is the length of the data (in addressable units) to be sent to the corresponding program. This value is 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.

datamd

(long:VALUE) This required parameter is the datamode to be used to send the connect data to the corresponding application. Refer to NRBDMODE in "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

SREAD C Function

The SREAD subroutine 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. 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 were shown in the SREAD function format. The parameters must be specified in the order presented. If parameters are omitted, then the commas must be used to preserve subsequent parameters' positions.

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

(struct nrb*) This required parameter is a pointer to the NRB data area to be passed to NETEX.

buffer

(extptr char *) This required parameter is the extended address of the buffer data area to receive the data sent by the corresponding application's SWRITE or SCONF request.

length

(long:VALUE) 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* input will be in NRBLEN. Programs that wish to work with the Unused Bit Count on input should examine the NRBUBIT field.

timeout

(long:VALUE) 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 were shown in the SWRIT function format. The parameters must be specified in the order presented. If parameters are omitted, then the commas must be used to preserve subsequent parameters' positions.

swrit
swritw

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

nrb

(struct nrb *) This required parameter is a pointer to the NRB data area to be passed to NETEX.

buffer

(extptr char *) This required parameter is the extended address of the buffer data area that holds the user data to be sent to the corresponding application.

length

(long:VALUE) 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.

datamd

(long:VALUE) This required parameter is the datamode to be used to send the connect data to the corresponding application. Refer to NRBDMODE in "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

SCLOS C Function

The SCLOS (close) function provides a means for a program to transmit data to another corresponding calling program and indicates 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 were shown in the SCLOS function format. The parameters must be specified in the order presented. If parameters are omitted, then the commas must be used to preserve subsequent parameters' positions.

sclos
sclosw

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

nrb

(struct nrb*) This required parameter is a pointer to the NRB data area to be passed to NETEX.

buffer

(extptr char*) This required parameter is the extended address of the pointer to the buffer data area that holds the user data to be sent to the corresponding application.

length

(long:VALUE) 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.

datamd

(long:VALUE) This required parameter is the datamode to be used to send the connect data to the corresponding application. Refer to NRBDMODE in "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
NRBBUFA	Contains zero

SWAIT C Function

The SWAIT function provides a means to wait for NETEX completions that were issued “nowaited.” The action taken by the subroutine will be different based on the value in the parameter NRBNUM.

If NRBNUM is greater than zero (swait specific) the subroutine will return control to the caller when any request identified in the NRB list has completed.

If NRBNUM is equal to zero, the subroutine will attempt to post all completed requests and return control to the caller immediately. If a previous call to SWAIT using NRBNUM equal to minus two (-2) has been issued but has not completed, the current SWAIT call will have no effect.

If NRBNUM is equal to minus one (-1), the subroutine will attempt to post all completed requests. If there were no completions at the time of the call, the subroutine will wait until a request has completed before returning control to the caller.

If NRBNUM is equal to minus two (-2), the subroutine attempts to post any single completed request and then immediately returns control to the caller If the call is made with the value parameter and there was a completion posted at the time of the call, then the value will contain the address of the NRB that completed. Otherwise, the value will contain zero.

SWAIT Function Format

The SWAIT function has the following format:

Function	**Parameters**
`swait`	`(nrbnum,nrb,nrb...,nrb10)`
`swait`	`(-2L)`

SWAIT Parameters

The following parameters were shown in the SWAIT function format. The parameters must be specified in the order presented. If parameters are omitted, then commas must be used to preserve subsequent parameters' positions.

swait

This is the verb for this function.

nrbnum

(long:VALUE) This required parameter is the number of NRBs in the nrb or one of the values described above.

nrb

(struct nrb*) This required parameter is an address pointer to one or more NRBs (the number of NRBs specified in nrbnum) associated with the request to be waited for (maximum of 10). An *nrb* is required for each NRB specified in *nrbnum*.

-2L

When this required parameter is specified, the subroutine attempts to post any single completed request and then immediately returns control to the caller.

The SWAIT call 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, after which it will return control to the user.

After control is returned, the calling C 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.

Note: While in an SWAIT (-2L) loop, do not do any waited session calls.

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 operator must wait for confirmation of previous SREAD or SWRIT calls 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 were shown in the SDISC function format. The parameters must be specified in the order presented. If parameters are omitted, then the commas must be used to preserve subsequent parameters' positions.

sdisc
sdiscw

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

nrb

(struct nrb*) This required parameter is a pointer to the NRB data area to be passed to NETEX.

buffer

(extptr char*) This required parameter is the extended address of the buffer data area that holds the user data to be sent to the corresponding application.

Note: In the single case of SDISC, delivery of DISCONNECT data is NOT reliable, although the actual disconnection will always occur.

length

(long:VALUE) This required parameter is the length of the data (in addressable units) to be sent to the corresponding program. This value is 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.

datamd

(long:VALUE) This required parameter is the datamode to be used to send the disconnect data to the corresponding application. Refer to NRBDMODE in “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

NETEX Utility Programs

The following utility programs are provided with the NRLUNIX 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 adapter where the local interface is connected. See "Step 6. Verify Operation" for more information.
cm	This is the Configuration Manager used to parse a text NCT file into a PAM file. CM is described in detail in the *"C" Configuration Manager and NETEX Path Retry (APR) User Guide*. See "Step 7. Review and Modify NCT File(s)" for more information.
nctl	This is a NETEX program that can be used to download a PAM file, created by CM, to a NESiGate LAN Offload adapter. See "NCT Loader Utility" for more information.

The following pages describe the *NTXOPER* and *NCTL* utility programs in more detail.

NETEX Operator Utilities

Overview

The operator interface is designed to allow limited manual control and display of NETEX resources such as remote LO NESiGate hardware, network adapters, remote hosts, or particular types of NETEX services. The NETEX operator facility accepts commands interactively.

The LO NESiGate provides two forms of operator interface through a session level connection. The two forms of operator interface are listed below:

- NETEX console interface
- NETEX remote operator interface

NETEX messages and operator commands are described in the Network Executive Software *NESiGate NetEx/IP Offload LAN-to-IP Gateway Reference Manual*.

NTXOPER - NETEX 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 NESiGate LAN Offload adapter on the local host system. 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.

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

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

```
$ ntxoper
```

The program will respond with the `ntxoper>` prompt, after which commands may be entered, for example:

```
ntxoper(host)> operatorcommand1
ntxoper(host)> operatorcommand2
```

NTXOPER commands may be entered either in upper or lowercase.

EXIT and QUIT Commands

The EXIT and QUIT commands are used to exit the NTXOPER program.

These commands have the following format:

Command (Select One)	**Parameters**
`EXit` `QUit`	

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 14.

Note: For a list of NESiGate LAN Offload adapter commands, enter a ?.

LOCAL Command

The LOCAL command selects the local mode and/or provide operator command information. In local mode, commands other than NTXOPER commands will be sent to the local NESiGate LAN Offload adapter 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`	`command`

LOCal

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

command

This optional parameter is a valid operator command for the local NETEX host.

REMOTE Command

The REMote command has the following format:

Command	**Required Parameters**	**Optional Parameters**
`REMote`	`hostname`	`command`

REMote

This is the verb for this command.

hostname

This required parameter is the remote host name identifier.

command

This optional parameter is a valid operator command for the remote NETEX host.

Examples

Commands that are not valid NTXOPER commands are automatically sent to the currently selected host-based 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 VAX may be obtained with the following NTXOPER command sequence:

```
$ NTXOPER
ntxoper(host)> REMote vax
VAX> d s
VAX> Exit
$
```

The first command in this sequence `(NTXOPER)` runs the NTXOPER utility program. Next, *REMOTE VAX* selects the remote host VAX. This is a valid NTXOPER command and thus, is interpreted by NTXOPER. *D S*, or Display Sessions, is not a valid NTXOPER command. Therefore, this command is passed by NTXOPER to the NETEX remote operator interface on the VAX host. Host VAX would then return a list of all currently active sessions to NTXOPER for display. Finally, the `EXIT` command terminates the NTXOPER program.

Figure 14 and Figure 15 shows an example program output from each of the following commands.

```
HELP
DISPLAY SESSIONS
```

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

```
$ NTXOPER
NTXOPER version x.x interactive local/remote NETEX operator
ntxoper> help
NTXOPER x.x command formats - (uppercase denotes required characters)
BYE               -- same as EXit
EXit                  exit NETEX operator
Help                  Briefly list available NTXOPER commands
LOCal                 switch to local command mode
LOCal cmd             execute local NETEX operator command
Quit                  -- same as exit --
REMote name           switch to remote command mode - host 'name'
REMote name cmd       execute host 'name' NETEX operator command
```

Figure 14. Screen Display: HELP Command (NTXOPER Utility)

```
$ NTXOPER
NTXOPER version x.x interactive local/remote NETEX operator
  ntxoper()> 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    CPBMO000
     NA        0    NA OFFERED    CPLPBR00
     NA        0    NA OFFERED    NTXNCTL0
     NA    20001    NA OFFERED    BFXJS
  ntxoper(VAX)> exit
  $
```

Figure 15. Screen Display: DISPLAY SESSION Command (NTXOPER Utility)

NCT Loader Utility

This chapter consists of two sections:

- NCTL Commands
- Configuration Parameters

NCTL Commands

The Network Control Table Loader is an interactive NETEX application program used for configuring local or remote NESiGate LAN Offload adapter. The NCT Loader takes a PAM file created by the Configuration Manager and transfers it to the NESiGate LAN Offload adapter through a NETEX connection.

Using the NCT Loader

The Interactive NCT Loader is started by entering the following command:

```
RUN DXNRL_CF:NCTL
```

On execution, the NCT Loader displays its name and version number, followed by a copyright notice.

When the command prompt `nctl:` appears, the operator may enter commands. These commands are described in the following sections. When multiple parameters are entered on a single line, the parameters are separated by one or more spaces. Commands may be entered as upper or lower case letters, and may be abbreviated to uniqueness (always the first letter). Character string parameters are used in the case (upper or lower) that they are entered. Numeric values are entered as decimal numbers (0-9). All entry lines are terminated by a carriage return. Operator entry follows the colon.

Note: For demonstration purposes, assume that there is a remote Apollo computer on the network. When using the NCT Loader on a real network, substitute the names for your systems.

Load NCT Command

This command loads a network configuration into a NESiGate LAN Offload adapter. If the hostname is omitted, the operator will be prompted for the hostname of the NESiGate LAN Offload adapter. If the filename is also omitted, the operator will be prompted for the filename of the PAMFILE. The default value for the hostname is the first eight characters of the PAMFILE, which the operator may accept with a carriage return. This is not always sufficient since the NESiGate LAN Offload adapter may not currently know itself by that name, and the local host (where the NCT Loader is running) may not know the NESiGate LAN Offload adapter by that name. The name the operator supplies must be the name by which the local host knows the relevant NESiGate LAN Offload adapter.

The LOAD command has the following format:

Command	Optional Parameters
`LOAD`	`filename hostname`

LOAD

This is the keyword for this command.

filename

This optional parameter is the name of the file containing the network configuration. This name may include the path to the file.

hostname

This optional parameter is the name of the host where the file will be loaded. The hostname can be from one to eight characters in length.

An actual entry could look like this:

```
nctl: load apollo.pam APOLLO
```

This takes the *apollo.pam* file as an PAM file and sends it to the APOLLO host. If the operator leaves out the parameters, the operator will be prompted first for the PAMFILE:

```
Enter PAMFILE name: apollo.pam
```

Once the PAMFILE filename is determined, the program will verify that the PAMFILE exists and determine its length. If the PAMFILE does not exist, the following message is displayed:

```
*** "filename" cannot be found
```

The program then returns to the command level. If the file length is less than one directory entry (16 bytes), the following message is displayed:

```
*** "filename" is too short
```

The program closes the file and returns to the command level. Otherwise, the operator is prompted for the hostname:

```
Enter hostname (APOLLO): APOLLO
```

The default hostname (in parentheses) is obtained from the first eight bytes of the PAMFILE. A carriage return without entry will use the default hostname. If the entered hostname is longer than eight characters, the following message is displayed:

```
filename90 is not a valid hostname.
```

If the load is successful, then the following message is displayed:

```
NCT Load SUCCESSFUL
```

If the load is not successful, the following message is displayed:

```
*** NCT Load failure, status dddd dddd
```

The two decimal numbers (dddd dddd) are status words returned by the NESiGate LAN Offload adapter. The successful/unsuccessful messages are only seen if no NETEX errors are encountered. Both messages are followed by:

```
  Number of BAD EEPROM cells found:           0
  Total number of EEPROM cells available: 15872
  Number of EEPROM cells used for NCT:     1058
```

The numbers shown in this example represents what might be displayed and indicate the usage of the EEPROM on the NESiGate LAN Offload adapter.

Help Command

This command displays help information about the NCT Loader's commands. Help for multiple commands may be displayed with a single entry, or a summary may be displayed.

The format for the HELP command is:

Command	Optional Parameters
`HELP`	`command...`

HELP

This is the keyword for this command.

command

This optional parameter is the name of one or more commands for which help is needed.

An actual entry could look like this:

```
nctl: help load quit xyz
```

If a command parameter does not exist (in the example, xyz), then the following message is displayed:

```
*** xyz is not a valid command.
```

If no command parameters are entered, then a summary of all the commands is displayed:

```
?       - To display command summaries and descriptions.
exit    - To exit the NCT Load Utility.
help    - To display command summaries and descriptions.
load    - To send a pamfile to a LO NESiGate adapter
quit    - To exit the NCT Load utility.
```

Figure 16. NCT Loader HELP Command Summary

Command parameters that are entered will display detailed information about the command (its syntax and a descriptive explanation).

The question-mark is also interpreted as a HELP command. All the commands currently are unique in their first character position. Therefore, all commands may be entered as a single character.

Quit/Exit Commands

These commands terminate the NCT Load program.

The format for the QUIT/EXIT command is:

Command (Select One)	Parameters
QUIT EXIT	

After entering one of the above commands, the following message is displayed and the program terminates.

```
        Completion of NCT Loader. Thank You.
```

Appendix A: 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 facilitate their subsequent examination by high level language programs.

NRBSTAT is designed to show whether an operation is in progress or 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 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, 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 specify the error type. The codes for error origin are listed below:

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 TCP/IP.
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.

The second digit (hundreds) places the errors in 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 time-outs
x4xx	NETEX-initiated disconnect errors
x5xx	Errors during connection

Note the following when using these codes:

- 0xxx and 90xx errors can be returned to any user program that accesses NETEX services. Normally, an application that accesses services at the transport level receives only those errors related to transport services (2xxx). 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 the transport level in the event of an adapter failure. If the transport level determines that this error should cause loss of communications, then the driver level (1xxx) error would be returned to the user with a Disconnect Indication in NRBIND when the next user read command was issued.
- 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.
- 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 further attempts to issue requests to that connection have an x100 (no N-ref) error returned to them.
- 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 subsections describe the errors in numerical order starting with general NETEX errors, followed by implementation-dependent, driver, session and network level errors.

General Errors

The following codes indicate general NETEX errors.

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.

0006 Invalid Odata Buffer Length. The buffer size specified in NRBPROTL 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.

0101* Invalid NREF. The user interface detected that the N-ref is not currently in use. The probable cause is a failing CONNECT or OFFER, or the failure to handle an incoming Disconnect.

0102* Invalid NREF (NRBREF ¼ SCBNREF). The user interface detected that the N-ref is not valid. The probable cause is an incorrectly modified NRB.

0103 Invalid address of the User Interface entry point within NRB. The user interface detected that the NRB is being used before a session has been established with an OFFER or CONNECT request. The probable cause is the lack of an OFFER or CONNECT or an incorrectly modified NRB.

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.

Implementation-dependent Errors (Socket Errors from TCP/IP)

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* (AAMFILE) 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* (AADESTADDRREQ) 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 TCP/IP.

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.

0980* (AANOCONN) No connection could be made. No server could connect 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 recv. 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.

0999* The interface could not allocate its extended segment.

Driver Level Errors

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 A DREAD request timed-out before any data was receive during this connection. The time value used for the time-out was in NRBTIME. No data was received. The status of the connection is not affected.

1310 Data that was receive during this connection has been discarded because the application did not issue a read request for a period of time greater than the data time-out period (30 seconds).**d** The status of the connection is not affected.

Session Level Errors

3300 A session level request (SREAD or SOFFER) timed-out before any data was receive during this session. The time value used for the time-out 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.

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 An SWRITE 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.

3307 An 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.

3310 Data that was received during this session has been discarded because the application did not issue a read request for a period of time greater than the data time-out period (30 seconds). The session is terminated.

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* The taskid for this SREF has been aborted by the host user interface.

3422* A 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 is 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 (SONNECT 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 SCONNECT 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 requested is not currently implemented.

3522* An offer terminated because of services drained.

3523* NETEX was DRAINed when a connect was received. This error is returned by the Session Manager to the connector.

Network Level 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 remain unchanged.

4101 In a Network connection that is intra-host (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 bitstream (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 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 overflow in this comparison will be allowed 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 remain unchanged.

4306 An NWRITE 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.

4307 An 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 the 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 returned from an SCONNECT request, it is because of an incorrectly-generated Network Configuration list.

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 a component of the network physically did not respond. The circuit cannot be established.

4513* During an attempt to establish a virtual circuit a component of the network could not honor the request because all of its circuit facilities were "busy." The circuit cannot be established at the current time.

4514* During an attempt to establish a virtual circuit a component of the network would not honor the request because of an equipment failure.

Index