



---

**NetEx/IP<sup>™</sup>**  
**for UNIX OR Windows Systems**

**Release 7.0.2-01**

---

**Software Reference Manual**



# Revision Record

Revision	Description
01 (08/2001)	Manual released.
02 (09/2002)	<ul style="list-style-type: none"><li>• Added additional NetEx operator commands.</li></ul>
03 (10/2003)	<ul style="list-style-type: none"><li>• Added additional NetEx/IP operator commands.</li><li>• Other minor revisions.</li></ul>
7.0 (03/2012)	<ul style="list-style-type: none"><li>• New Licensing support</li><li>• Major updating of manual</li></ul>
7.0.1 (06/2013)	<ul style="list-style-type: none"><li>• Minor corrections</li><li>• Update to support Linux for System z</li><li>• Update to support AIX</li></ul>
7.0.1-01	<ul style="list-style-type: none"><li>• Add GPG signature to distributed packages</li></ul>
7.0.1-02	<ul style="list-style-type: none"><li>• Add 7.0.1 support for Linux on x86 platforms</li></ul>
7.0.2	<ul style="list-style-type: none"><li>• Add 7.0.2 support for Windows on x86 platforms and Solaris on Sparc platforms</li></ul>
7.0.2-01	<ul style="list-style-type: none"><li>• Correct comment characters in NESikeys file; update company address</li></ul>

© 2004-2017 by Network Executive Software. Reproduction is prohibited without prior permission of Network Executive Software. Printed in the U.S.A. All rights reserved.

You may submit written comments to:

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

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

[support@netex.com](mailto:support@netex.com)

or, by visiting our web site at:

<http://www.netex.com>

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



# Preface

This manual describes the NetEx/IP™ software for supported UNIX or Windows operating systems.

The supported UNIX/Windows operating systems and specific NetEx/IP™ products are:

- H620IP for IBM AIX operating systems.
- H800IP for Linux operating system on x86 platforms.
- H820IP for Linux for System z operating systems.
- H140IP for Window Operating Systems on x86 platforms.
- H690IP for Solaris Operating Systems on Sparc platforms.

“Chapter 1: Introduction”, “Chapter 2: NetEx/IP and the ISO Model”, “Chapter 3: NetEx/IP Session Services”, and “Chapter 4: NetEx Request Block” are intended for all readers. “Chapter 5: C High Level Interface” describes the library of subroutines that are called by the C high-level language programs.

“Appendix A: NRB Error Codes” includes a list and description of the error messages and codes issued by NetEx/IP.

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

# Reference Material

The following manuals contain related information.

<b>Number</b>	<b>Title and Description</b>
MAN-CNET-CONF-MGR	<i>"C" Configuration Manager and NETEX Alternate Path Retry (APR) User Guide</i>

# 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 that are 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

<b>Network Executive Software</b>	<b>NetEx, NetEx/IP, BFX, PFX, USER-Access, eFT</b>
<b>Hewlett-Packard Company</b>	<b>HP, HP-UX</b>
<b>The Open Group</b>	<b>UNIX</b>
<b>International Business Machines</b>	<b>IBM, AIX, RISC 6000, RS/6000, System z</b>
<b>Microsoft Corporation</b>	<b>Windows</b>
<b>SUN Microsystems, Inc.</b>	<b>SUN, Solaris</b>
<b>Red Hat, Inc.</b>	<b>Red Hat</b>
<b>Linus Torvalds</b>	<b>Linux</b>

These references are made for informational purposes only.

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

## Notice to the Customer

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

# Document Conventions

The following notational conventions are used in this document.

Format	Description
displayed information	Information displayed on a CRT (or printed) is shown in <i>this font</i> .
<i>user entry</i>	<i>This font</i> 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.
<b>bold</b>	The exact form of a keyword that is case-sensitive and all or part of it must be issued in lowercase.
lowercase	A user-supplied name or string.
value	Underlined parameters or options are defaults.
<label>	The label of a key appearing on a keyboard. If "label" is in uppercase, it matches the label on the key (for example: <ENTER>). If "label" is in lowercase, it describes the label on the key (for example: <up-arrow>).
<key1><key2>	Two keys to be pressed simultaneously.
No delimiter	Required keyword/parameter.



# Glossary

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

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

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

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

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

**Coprocessor NETwork EXecutive (CP NetEx):** Resides on some types of Processor Interface (PI) boards and uses the processing and storage capacity of the board. This allows minicomputer users to use NetEx with minimal impact on host storage and processing.

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

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

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

**ISO:** Acronym for International Standards Organization.

**link:** (1) A joining of any kind of IP networks. (2) The communications facility used to interconnect two trunks/busses on a network.

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

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

NetEx can reside on the host or in a NESiGate adapter.

NetEx is a registered trademark of Network Executive Software.

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

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

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

**TCP/IP:** An acronym for Transmission Control Protocol/Internet Protocol. These communication protocols provide the mechanism for inter-network communications, especially on the Internet. The protocols are

hardware-independent. They are described and updated through *Requests For Comment* (RFC). IP corresponds to the OSI network layer 3, TCP to layers 4 and 5.

# Contents

<b>Revision Record .....</b>	<b>i</b>
<b>Preface.....</b>	<b>iii</b>
<b>Reference Material.....</b>	<b>iv</b>
<b>Notice to the Reader.....</b>	<b>v</b>
Corporation Trademarks and Products.....	v
Notice to the Customer .....	v
Document Conventions.....	vi
Glossary .....	vii
<b>Contents .....</b>	<b>ix</b>
Figures.....	xx
Tables.....	xxi
<b>Chapter 1: Introduction .....</b>	<b>1</b>
NetEx/IP Characteristics .....	1
External Interface.....	1
Internal Interaction.....	1
NetEx/IP Connections.....	1
Design Efficiency and Flexibility .....	2
Block Segmenting.....	2
Alternate Path Retry.....	2
Remote Operator Interface.....	2
New Features .....	2
Basic I/O Flow .....	3
Host Based NetEx .....	3
Host Based NetEx/IP .....	3
NESiGate / NetEx/IP .....	3
<b>Chapter 2: NetEx/IP and the ISO Model.....</b>	<b>5</b>
Session Layer Services.....	6
Transport Layer Services .....	6
Network Layer Services.....	7
Driver Sublayer Services .....	7
<b>Chapter 3: NetEx/IP Session Services.....</b>	<b>9</b>
Session Layer Requests.....	9
General Concept of a Session .....	10
Establishing a Session.....	12
Data Transfer.....	14
Write/Read Data Transfer .....	14
Concurrent Write and Read Data Transfer .....	16
One-Way Data Transfer.....	18
Terminating A Session.....	20
Normal Termination .....	20

Abnormal Session Termination.....	21
Programming Notes.....	21
Handling Multiple Connections .....	21
Service WAIT Options.....	22
Satellite Communication .....	22
NetEx/IP Error Recovery Procedures.....	23
Error Codes.....	23
Common Error Recovery Procedures.....	23
Code Conversion.....	24
<b>Chapter 4: NetEx Request Block .....</b>	<b>25</b>
Rules for NRB Usage.....	25
NRB Fields .....	25
NRBSTAT.....	26
NRBIND.....	27
NRBLEN and NRBUBIT.....	27
NRBREQ.....	28
NRBNREF .....	29
NRBBUFA .....	29
NRBBUFL .....	29
NRBDMODE.....	29
Manual Datamode .....	29
Auto Datamode.....	30
NRBTIME.....	31
NRBCLASS .....	32
NRBMAXRT .....	32
NRBBLKI and NRBBLKO.....	32
NRBPROTA and NRBPROTL.....	33
NRBRESV1 and NRBRESV2 .....	34
NRBOFFER .....	34
NRBHOST .....	34
NRBRESV3 .....	34
NRBUSER .....	34
NRBOSD.....	34
Creating an NRB .....	34
Duplicating an NRB .....	34
<b>Chapter 5: C High Level Interface .....</b>	<b>37</b>
C NETEX Request Blocks .....	37
SOFFR C Function .....	39
SOFFR Function Format .....	39
SOFFR Parameters .....	39
SOFFER Entry Parameters.....	40
SOFFR Results .....	40
SCONN C Function.....	41
SCONN Function Format.....	41
SCONN Parameters.....	41
SCONN Entry Parameters.....	42
SCONN Results.....	42
SCONF C Function .....	43
SCONF Function Format .....	43

SCONF Parameters.....	43
SCONF Entry Parameters.....	43
SCONF Results.....	44
SREAD C Function.....	45
SREAD Function Format.....	45
SREAD Parameters.....	45
SREAD Entry Parameters.....	46
SREAD Results.....	46
SWRIT C Function.....	47
SWRIT Function Format.....	47
SWRIT Parameters.....	47
SWRIT Entry Parameters.....	47
SWRIT Results.....	48
SCLOS C Function.....	49
SCLOS Function Format.....	49
SCLOS Parameters.....	49
SCLOS Entry Parameters.....	49
SCLOS Results.....	50
SWAIT C Function.....	51
C SWAIT Examples.....	52
SWAIT Function Format.....	53
SWAIT Parameters.....	53
SDISC C Function.....	54
SDISC Function Format.....	54
SDISC Parameters.....	54
SDISC Entry Parameters.....	55
SDISC Results.....	55
C Program Examples.....	56
<b>Chapter 6: TNP.....</b>	<b>61</b>
Overview.....	61
TNP Configuration.....	62
NetEx/IP License Key.....	62
TNP PROGRAM.....	62
tnp.cfg.....	63
tnpop.....	63
Adding NetEx/IP Requester Hosts to the NCT.....	64
Requester Configuration.....	65
<b>Chapter 7: Installation.....</b>	<b>67</b>
<b>Chapter 8: Operator Interface.....</b>	<b>69</b>
Executing Commands.....	70
Command Line Mode.....	70
Interactive Mode.....	70
Enabling the Remote Operator Service.....	71
Operator Command Descriptions.....	72
CLEAR LOG.....	74
Description.....	74
Format.....	74
CLEAR IPRROUTE.....	75

Description .....	75
Format .....	75
Parameters .....	75
DISPLAY DRAINED .....	76
Description .....	76
Format .....	76
DISPLAY HOST .....	77
Description .....	77
Format .....	77
Parameters .....	77
Display Examples .....	77
DISPLAY IPROUTE .....	79
Description .....	79
Format .....	79
Parameters .....	79
Display Examples .....	79
DISPLAY KEY .....	81
Description .....	81
Format .....	81
Display Examples .....	81
DISPLAY LOG .....	82
Description .....	82
Format .....	82
DISPLAY MEMORY .....	83
Description .....	83
Format .....	83
Display Examples .....	83
DISPLAY NETWORK .....	85
Description .....	85
Format .....	85
Parameters .....	85
Display Examples .....	85
DISPLAY PARMS .....	87
Description .....	87
Format .....	87
Display Examples .....	87
DISPLAY SESSION .....	93
Description .....	93
Format .....	93
Parameters .....	93
Display Examples .....	93
DISPLAY TRANSPORT .....	95
Description .....	95
Format .....	95
Parameters .....	95
Display Examples .....	95
DISPLAY USAGE .....	101
Description .....	101
DISPLAY VERSION .....	102
Description .....	102
Format .....	102

Display Example .....	102
DRAIN ADAPTER .....	102
Description .....	102
Format .....	102
Parameters .....	102
DRAIN NETEX .....	103
Description .....	103
Format .....	103
DRAIN HOST .....	104
Description .....	104
Format .....	104
Parameters .....	104
HALT SREF .....	105
Description .....	105
Format .....	105
Parameters .....	105
HELP and lhelp .....	106
Description .....	106
Format .....	106
lhelp .....	106
Parameters .....	106
Comments .....	106
KILL NETEX .....	107
Description .....	107
Format .....	107
LOAD KEY .....	108
Description .....	108
Format .....	108
LOAD NCT .....	109
Description .....	109
Format .....	109
Parameters .....	109
SET BUFOLIM .....	110
Description .....	110
Format .....	110
Parameters .....	110
SET CONTIME .....	111
Description .....	111
Format .....	111
Parameters .....	111
SET DBGDATA .....	112
Description .....	112
Format .....	112
Parameters .....	112
SET DBGMSG .....	113
Description .....	113
Format .....	113
Parameters .....	113
SET DBGREQ .....	114
Description .....	114
Format .....	114

Parameters .....	114
SET DBGRET .....	115
Description .....	115
Format .....	115
Parameters .....	115
SET DEADTIME .....	116
Description .....	116
Format .....	116
Parameters .....	116
SET DEFBI .....	117
Description .....	117
Format .....	117
Parameters .....	117
SET DEFBO .....	118
Description .....	118
Format .....	118
Parameters .....	118
SET IDLETIME .....	119
Description .....	119
Format .....	119
Parameters .....	119
SET IPRROUTE .....	120
Description .....	120
Format .....	120
Parameters .....	120
Notes .....	120
SET MAXBI .....	121
Description .....	121
Format .....	121
Parameters .....	121
SET MAXBO .....	122
Description .....	122
Format .....	122
Parameters .....	122
SET MAXKBS .....	123
Description .....	123
Format .....	123
Parameters .....	123
SET MBUFIN .....	124
Description .....	124
Format .....	124
Parameters .....	124
SET MBUFOUT .....	125
Description .....	125
Format .....	125
Parameters .....	125
SET MSGLVL .....	126
Description .....	126
Format .....	126
Parameters .....	126
SET MSGSYSLOG .....	127



Description.....	127
Format.....	127
Parameters.....	127
SET MSGSYSLOGFAC .....	128
Description.....	128
Format.....	128
Parameter .....	128
SET MULTHOST .....	130
Description.....	130
Format.....	130
Parameters.....	130
SET NTXOPER.....	131
Description.....	131
Format.....	131
Parameters.....	131
SET POLLSEL .....	132
Description.....	132
Format.....	132
Parameters.....	132
SET PREFPROT.....	133
Description.....	133
Format.....	133
Parameters.....	133
SET RCVDATAQHB.....	134
Description.....	134
Format.....	134
Parameters.....	134
SET RCVDATAQLB .....	135
Description.....	135
Format.....	135
Parameters.....	135
SET RCVDATAQHS .....	136
Description.....	136
Format.....	136
Parameters.....	136
SET RCVDATAQLS .....	137
Description.....	137
Format.....	137
Parameters.....	137
SET REXMWBLKS.....	138
Description.....	138
Format.....	138
Parameters.....	138
SET READTIME.....	139
Description.....	139
Format.....	139
Parameter .....	139
SET ROPCLASS .....	140
Description.....	140
Format.....	140
Parameters.....	140

SET SESMAX.....	141
Description .....	141
Format .....	141
Parameters .....	141
SET USERCVGAPQ.....	142
Description .....	142
Format .....	142
Parameters .....	142
SET USEREXMITQ.....	143
Description .....	143
Format .....	143
Parameters .....	143
SET WDOGINT.....	144
Description .....	144
Format .....	144
Parameters .....	144
START NETEX.....	144
Description .....	144
Format .....	144
START ADAPTER.....	145
Description .....	145
Format .....	145
Parameters .....	145
START HOST.....	146
Description .....	146
Format .....	146
Parameters .....	146
<b>Appendix A: NRB Error Codes .....</b>	<b>147</b>
General Errors .....	148
Host Specific Errors.....	150
License Specific Errors.....	150
Driver Service Errors.....	150
Transport Service Errors.....	153
Session Service Errors .....	155
Network Service Errors .....	158
<b>Appendix B: Messages .....</b>	<b>161</b>
Messages: .....	162
<b>Appendix C: Running Multiple NetExes.....</b>	<b>169</b>
Atypical Operations.....	169
Required Shell Environment Variables .....	169
Required ntx_default File Parameter Settings.....	169
Suggested ntx_default File .....	169
Order of Events For Multiple NetEx Instances On a Server: .....	169
Example of Two Netexes on a Server .....	170
Netex1 .....	171
Netex2 .....	171
Other Considerations.....	172

<b>Appendix D: H800IP Linux Installation</b> .....	<b>173</b>
Prerequisites .....	173
Hardware Installation.....	173
Accessing the H800IP software distribution.....	173
Getting the NESi Public Key .....	173
Importing the NESi Public Key .....	173
Verifying Signatures .....	173
Software Installation .....	174
Upgrading H800IP .....	174
Removing H800IP .....	174
Removing the NESi Public Key .....	174
Starting, Stopping & Verifying Install of Netex .....	174
Post Installation Considerations.....	175
Configuring H800IP.....	175
Step 1. Edit the ‘NESikeys’ file .....	175
Step 2. Edit the NTX_DEFAULT file .....	176
Step 3. Build an NCT .....	176
Step 4. Create NetEx/IP addressing information .....	177
Step 5. Create Code Conversion Table (optional).....	177
Step 6. Start NetEx.....	177
Step 7. Verify that ‘ntx’ Starts Automatically On Reboot .....	178
Step 8. Setup syslog file definition & logrotate rules .....	178
<b>Appendix E: H620IP AIX Installation</b> .....	<b>179</b>
Prerequisites .....	179
Hardware Installation.....	179
Accessing the H620IP software distribution.....	179
Software Installation .....	179
Upgrading H620IP if RPM was not used for the installation .....	179
Upgrading H620IP if RPM was used to install.....	180
Removing H620IP RPM.....	180
Starting, Stopping & Verifying Install of NetEx .....	180
Post Installation Considerations.....	181
Configuring H620IP.....	181
Step 1. Edit the ‘NESikeys’ file .....	181
Step 2. Edit the NTX_DEFAULT file .....	182
Step 3. Build an NCT .....	182
Step 4. Create NetEx/IP addressing information .....	182
Step 5. Create Code Conversion Table (optional).....	183
Step 6. Starting / Stopping NetEx .....	183
Step 7. Verify that ‘ntx’ Starts Automatically On Reboot .....	184
Step 8. Setup syslog file definition & logrotate rules (optional).....	184
<b>Appendix F: H820IP Linux for System z Installation</b> .....	<b>185</b>
Prerequisites .....	185
Hardware Installation.....	185
Accessing the H820IP software distribution.....	185
Getting the NESi Public Key .....	185
Importing the NESi Public Key .....	185
Verifying Signatures .....	185
Software Installation .....	186

Upgrading H820IP .....	186
Removing H820IP .....	186
Removing the NESi Public Key .....	186
Starting, Stopping & Verifying Install of Netex.....	187
Post Installation Considerations .....	188
Configuring H820IP .....	188
Step 1. Edit the 'NESikeys' file.....	188
Step 2. Configuring the default file .....	189
Step 3. Build an NCT .....	189
Step 4. Create NetEx/IP addressing information.....	190
Step 5. Create Code Conversion Table (optional).....	190
Step 6. Start NetEx .....	191
Step 7. Verify that 'ntx' Starts Automatically On Reboot.....	191
Step 8. Setup syslog file definition & logrotate rules .....	191
<b>Appendix G: H140IP Window on Intel x86 Platforms Installation .....</b>	<b>193</b>
Prerequisites .....	193
Hardware Installation .....	193
Accessing the H140IP software distribution .....	193
Software Installation.....	193
Upgrading H140IP .....	193
Removing H140IP.....	193
Starting and Stopping NetexIP .....	193
Post Installation Considerations .....	194
Configuring H140IP .....	194
Step 1. Edit the 'NESikeys' file.....	194
Step 2. Configuring the default file .....	195
Step 3. Build an NCT .....	195
Step 4. Create NetEx/IP addressing information.....	196
Step 5. Create Code Conversion Table (optional).....	196
Step 6. Start NetEx .....	197
Step 7. Verify that 'ntx' Starts Automatically On Reboot.....	197
<b>Appendix H: H690IP Solaris on Sparc Installation .....</b>	<b>199</b>
Prerequisites .....	199
Hardware Installation .....	199
Accessing the H690IP software distribution .....	199
Upgrading H690IP .....	199
Removing H690IP.....	199
Starting, Stopping & Verifying Install of Netex/IP .....	200
Software Installation.....	200
Configuring H690IP .....	201
Step 1. Edit the 'NESikeys' file.....	201
Step 2. Configuring the default file .....	202
Step 3. Build an NCT .....	202
Step 4. Create NetEx/IP addressing information.....	203
Step 5. Create Code Conversion Table (optional).....	203
Step 6. Start NetEx .....	204
Step 7. Verify that 'ntx' Starts Automatically On Reboot.....	204
Step 8. Setup syslog file definition & logrotate rules .....	204

<b>Appendix I: NetEx Default File Configuration .....</b>	<b>205</b>
Edit the NTX_DEFAULT file .....	205
<b>Appendix J: NetEx Default Parameters Mapping .....</b>	<b>215</b>
<b>Index.....</b>	<b>219</b>

# Figures

Figure 1. Basic I/O Flow .....	3
Figure 2. ISO Model Communication .....	5
Figure 3. NetEx/IP and the ISO Model .....	6
Figure 4. Sample System Configuration.....	11
Figure 5. Simplified Session Example .....	11
Figure 6. Establishing a Connection.....	13
Figure 7. Write/Read Data Transfer .....	15
Figure 8. Concurrent SREAD and SWRITE Requests.....	17
Figure 9. One-Way Data Transfer .....	19
Figure 10. Normal Session Termination.....	20
Figure 11. NetEx Request Block (NRB) Words.....	26
Figure 12. SWAIT(0) Example .....	52
Figure 13. SWAIT(-1) Example.....	53
Figure 14. C Program Example .....	60
Figure 15. Output display of an OFFERED TNP job.....	61
Figure 16. Output display of connected TNP jobs .....	62
Figure 17. Requester host configuration file .....	63
Figure 18. Sample NCT with NetEx/IP Requester host .....	64
Figure 19. Sample NetEx/IP Requester configuration .....	65
Figure 20. NETEX Operator Commands .....	73
Figure 21. Display Hosts Command Output, No HostName specified. ....	77
Figure 22. Display Host Command, HostName specified.....	78
Figure 23. Display IP Route Command .....	79
Figure 24. Display Key Command.....	81
Figure 25. Display Memory Command.....	83
Figure 26 Display Network Command.....	85
Figure 27. Display Network Command, NREF specified .....	85
Figure 28. Display Parms Command.....	87
Figure 29. Display Session, no SREF specified .....	93
Figure 30. Display Session, SREF specified .....	93
Figure 31. Display Transport, no TREF specified.....	95
Figure 32. Display Transport, TREF specified.....	96

Figure 33. Display Version Command .....	102
--	-----

## Tables

Table 1. ISO Model.....	5
Table 2. Origin of NRB Error Codes .....	147
Table 3. Origin of NRB Error Codes, Part 2.....	148
Table 4. General NRB Error Codes .....	149
Table 5. Host Specific NRB Error Codes .....	150





# Chapter 1: Introduction

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

## NetEx/IP Characteristics

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

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

### External Interface

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

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

### Internal Interaction

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

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

### NetEx/IP Connections

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

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

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

## Design Efficiency and Flexibility

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

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

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

## Block Segmenting

NetEx/IP products provide block segmenting at the transport layer. NetEx/IP divides data into segments of a specified size for transmission across the network and reassembles the segments on the remote host before delivering the data to the session layer calling program on the remote NetEx/IP. This segmenting is transparent to the session user but provides control of the transmitted block segment size. This is especially useful for 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/IP versions. For more information on APR, refer to the *“C” Configuration Manager and NETEX Alternate Path Retry (APR) User Guide*.

## Remote Operator Interface

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

## New Features

NetEx/IP release 7.0 continues to provide support for standard IP networks that was introduced in Release 5.0. Release 6.0 introduced enhancements to this support by introducing NetEx/IP protocol extensions (referred to as Type 4 protocol), that provide the ability for NetEx/IP to dynamically maximize the network performance, based on factors such as available bandwidth, distance, and workload on the network. (This protocol type is not yet supported.)

## Basic I/O Flow

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

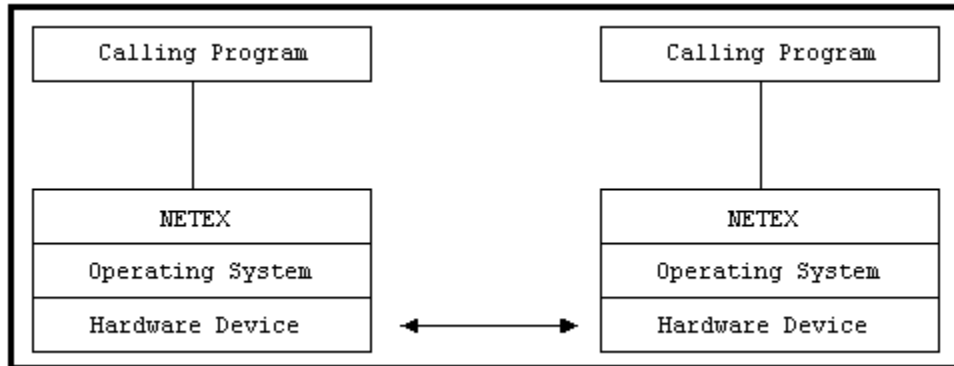


Figure 1. Basic I/O Flow

## Host Based NetEx

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/IP

NetEx/IP release 7.0 provides support for communicating over standard IP networks. No changes are required to existing NetEx applications, since the Application Programming Interfaces (API's) have not been changed.

## NESiGate / NetEx/IP

The NetEx/IP 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 the Compaq OpenVMS TCP/IP product, or H367IP the HP Guardian TCPIP product, is used for transport of NetEx requests and buffers between the H267IP (or H367IP) host and the NESiGate adapter or the TNP NetEx.



# Chapter 2: NetEx/IP and the ISO Model

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

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

Table 1. ISO Model	
Layer	Major Functions
Application	High level description of data to be transferred and the destination involved
Presentation	Select data formats and syntax
Session	Establish session connection, report exceptions, and select routing
Transport	Manage data transfer and provide NetEx/IP-to-NetEx/IP message delivery
Network	Point-to-point transfer, error detection, and error recovery
Data Link	Data link connection, error checking, and protocols
Physical	Mechanical and electrical protocols and interfaces

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

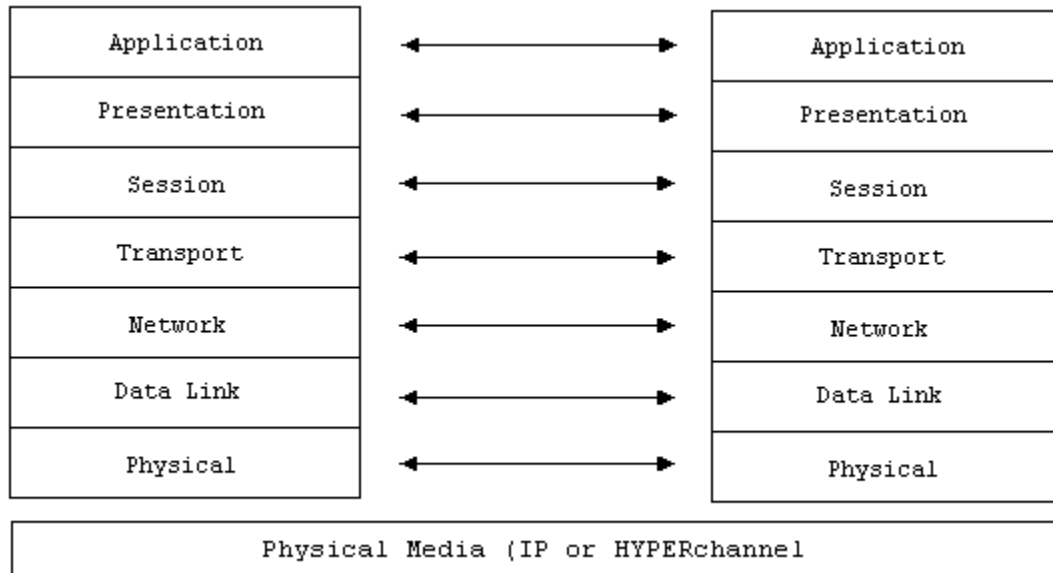


Figure 2. ISO Model Communication

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

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

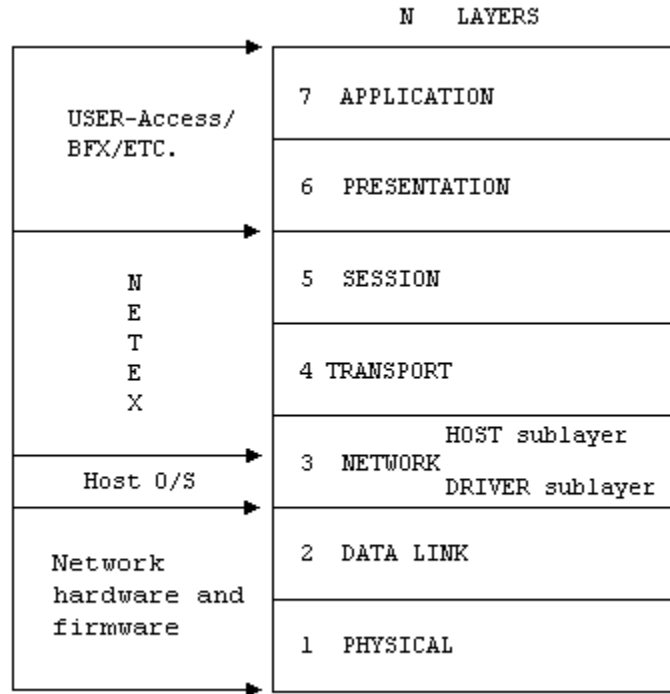


Figure 3. NetEx/IP and the ISO Model

## Session Layer Services

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

## Transport Layer Services

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

The transport software manages the network path chosen by the session software. The session user does not need to be concerned with the actual hardware and software used to transmit data, nor with NetEx/IP-to-NetEx/IP message delivery. The transport layer sets up hardware and software tables, provides buffering, and establishes linkages to manage the flow of information. Also, the protocol used by the transport layer software provides true full-duplex communications between subsystems, permitting asynchronous reads and writes. Because the transport layer provides a full-duplex operation, data can flow continuously, as long as it is being supplied by the user. This keeps the 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/IP and assumes responsibility for keeping the network interfaces busy. This is an internal layer used only by the internal transport service, not the end user. The network layer formats the message 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.





# Chapter 3: NetEx/IP Session Services

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

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

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

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

## Session Layer Requests

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

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

### **SOFFER**

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

### **SCONNECT**

This command requests a session with a calling program that previously issued an SOFFER. The program may insert values defining characteristics of the upcoming session in the NRB when this re-

quest is issued. This request may also write data to the offering program provided this data does not exceed the maximum segment size specified on either the sending or receiving NetEx/IP.

### **SCONFIRM**

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

### **SWRITE**

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

### **SREAD**

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

### **SWAIT**

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

### **SCLOSE**

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

### **SDISCONNECT**

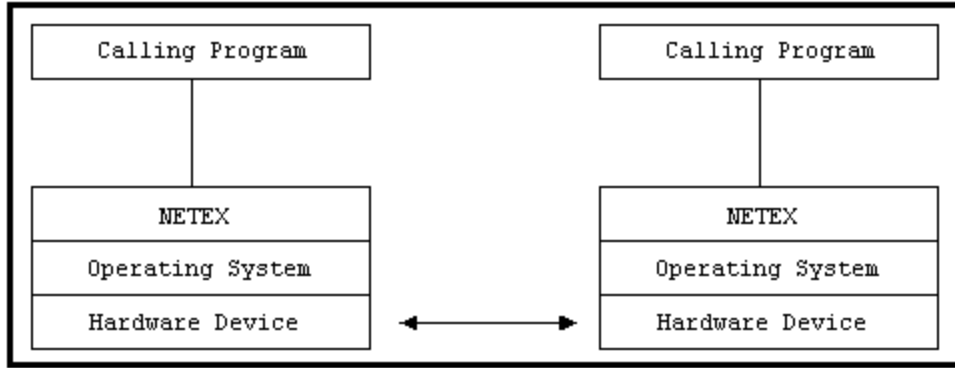
This command immediately terminates a connected session. The SDISCONNECT request may be issued by either program at any time.

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

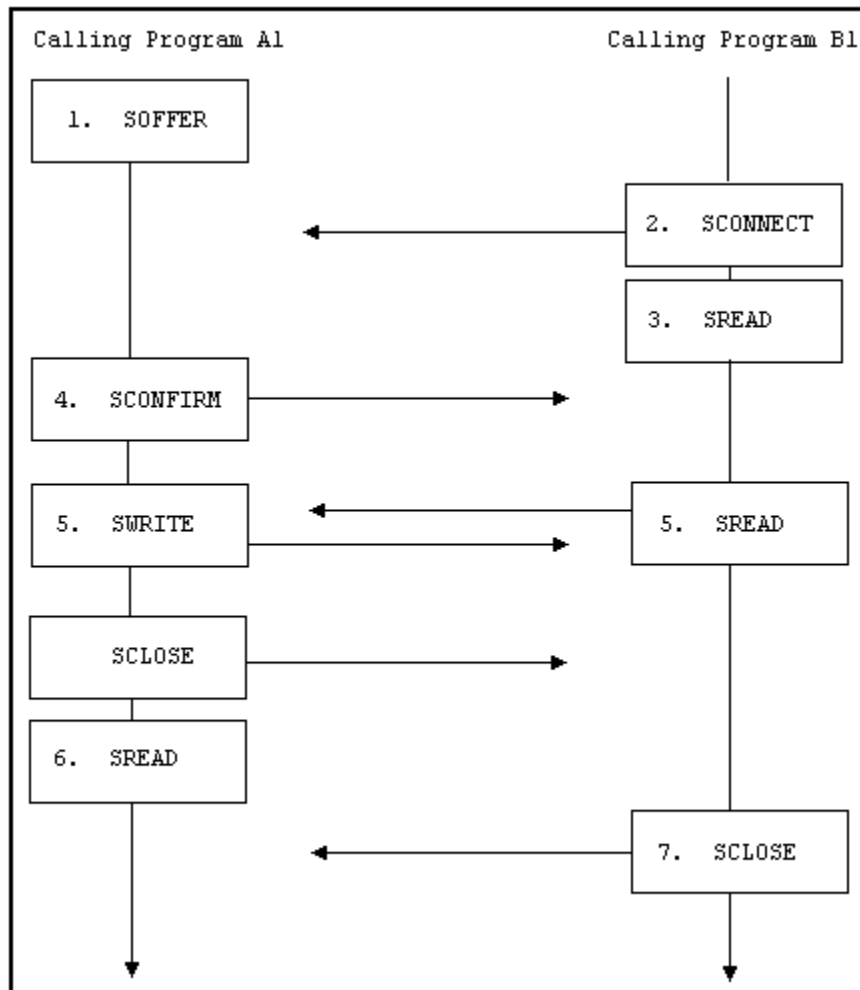
## **General Concept of a Session**

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

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



**Figure 4. Sample System Configuration**



**Figure 5. Simplified Session Example**

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

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

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

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

The SREAD that program B1 previously issued now indicated program A1's response. If program A1 issued a SCONFIRM, the SREAD will show a Confirm indication along with the data sent with the SCONFIRM. If program A1 issued an SDISCONNECT, the SREAD will complete with a Disconnect indication.

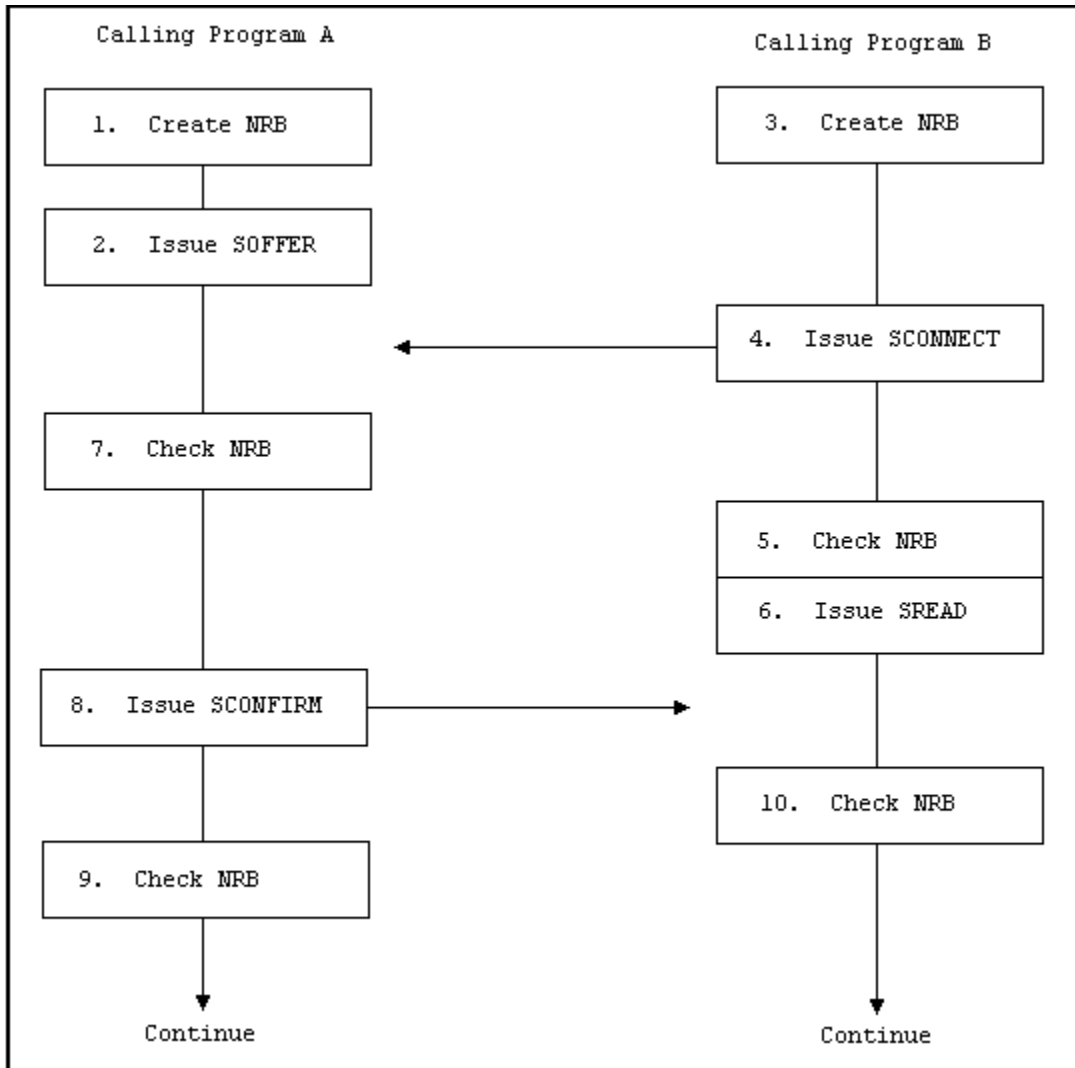
5. The programs may now begin the data transfer. Program B1 issues an SREAD to prepare to receive data from program A1. Program A1 writes data to program B1. The SWRITE command is used until the last data is written. An SCLOSE is used to write the last data. Since we are only issuing one write request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

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

## Establishing a Session

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

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



**Figure 6. Establishing a Connection**

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

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

Figure 6 continues assuming the NRB indicated normal completion. If the NRB indicates that a request is in progress, it would have to be rechecked after a short delay. If the NRB indicates an error, the error code would be logged and the session would not be established. The calling program should then either

try again to establish a session, or should act appropriately (such as closing files that were opened before the session was attempted).

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

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

A password may be required at this time. The password could be sent as data associated with the SCONNECT. After the SCONNECT is received, the password is examined. The password could be used to restrict access to certain files, restrict access by certain programs, or have other customized uses. If a program attempts to access restricted files or has an incorrect password, program A may issue an SDISCONNECT and terminate the session.

8. If all conditions associated with the SCONNECT are acceptable, program A issues an SCONFIRM to establish the session. The SCONFIRM may contain data.
9. Program A checks the NRB to make sure that the SCONFIRM was accepted by NetEx/IP. If it was, program A would continue with the session. If NetEx/IP did not accept the SCONFIRM, program A would either retry issuing the SCONFIRM or take other appropriate action.
10. Program B checks the NRB to determine if the SREAD has successfully completed and to see what call program A issued. If program A had responded with an SCONFIRM, program B would continue with the session. If program A had responded with an SDISCONNECT, program B would have to examine the reason code associated with the SDISCONNECT, and take the appropriate action.

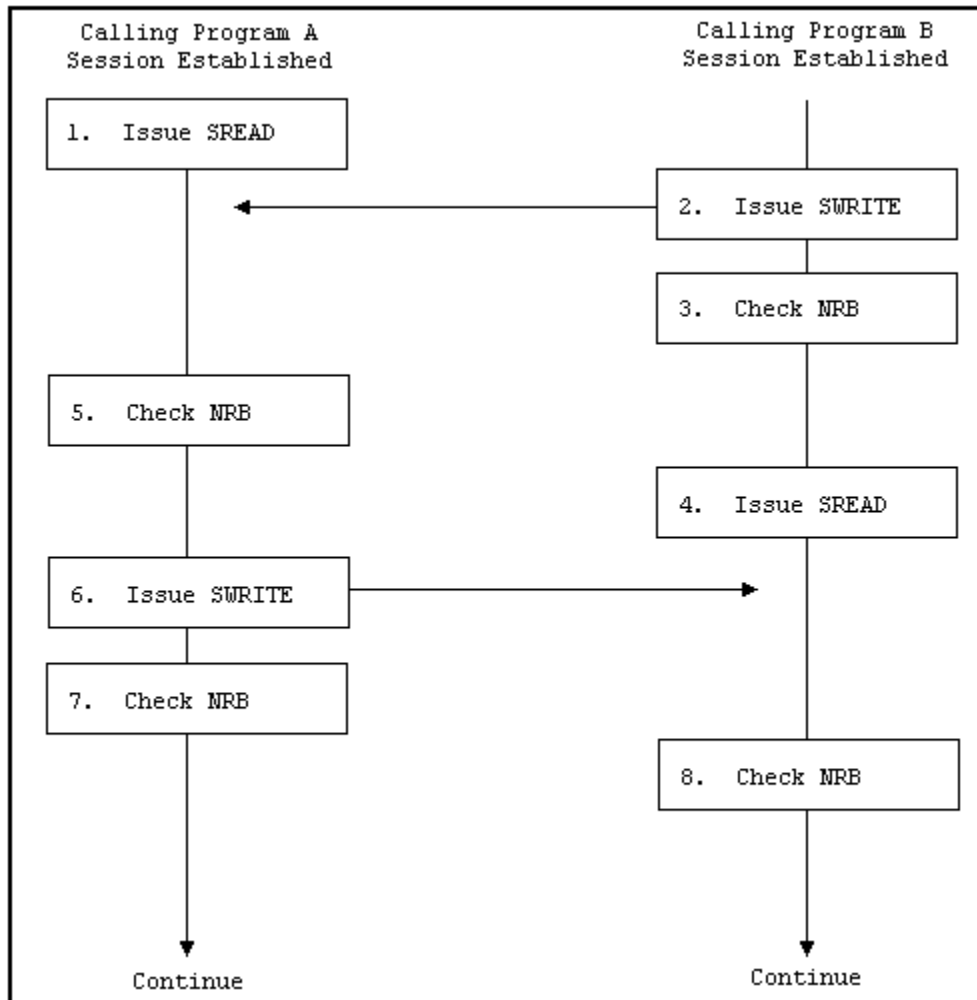
The previous discussion outlines the rules to be followed when establishing a NetEx/IP session. It also introduces the concept of using the NRB for communication with NetEx/IP. After requests are issued, the NRB is examined to see when NetEx/IP completes the request. This may be done using the SWAIT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

## Data Transfer

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

### Write/Read Data Transfer

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



**Figure 7. Write/Read Data Transfer**

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

1. After a session has been established, program A issues an SREAD. The SREAD specifies the buffer for receiving data.
2. Program B issues an SWRITE. The SWRITE specifies where the data to be written is located and how long it is.
3. Program B checks the NRB to see if NetEx/IP accepted the SWRITE or indicated an error.  
Once NetEx/IP has accepted the data, it will deliver the data unless a catastrophic loss of connection occurs.
4. Program B issues an SREAD to detect program A's next request.
5. Program A received an updated NRB which indicates what program B has issued. In this case, program B has written data as program A expected. If the NRB word indicated an error, program A would act appropriately.

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

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

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

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

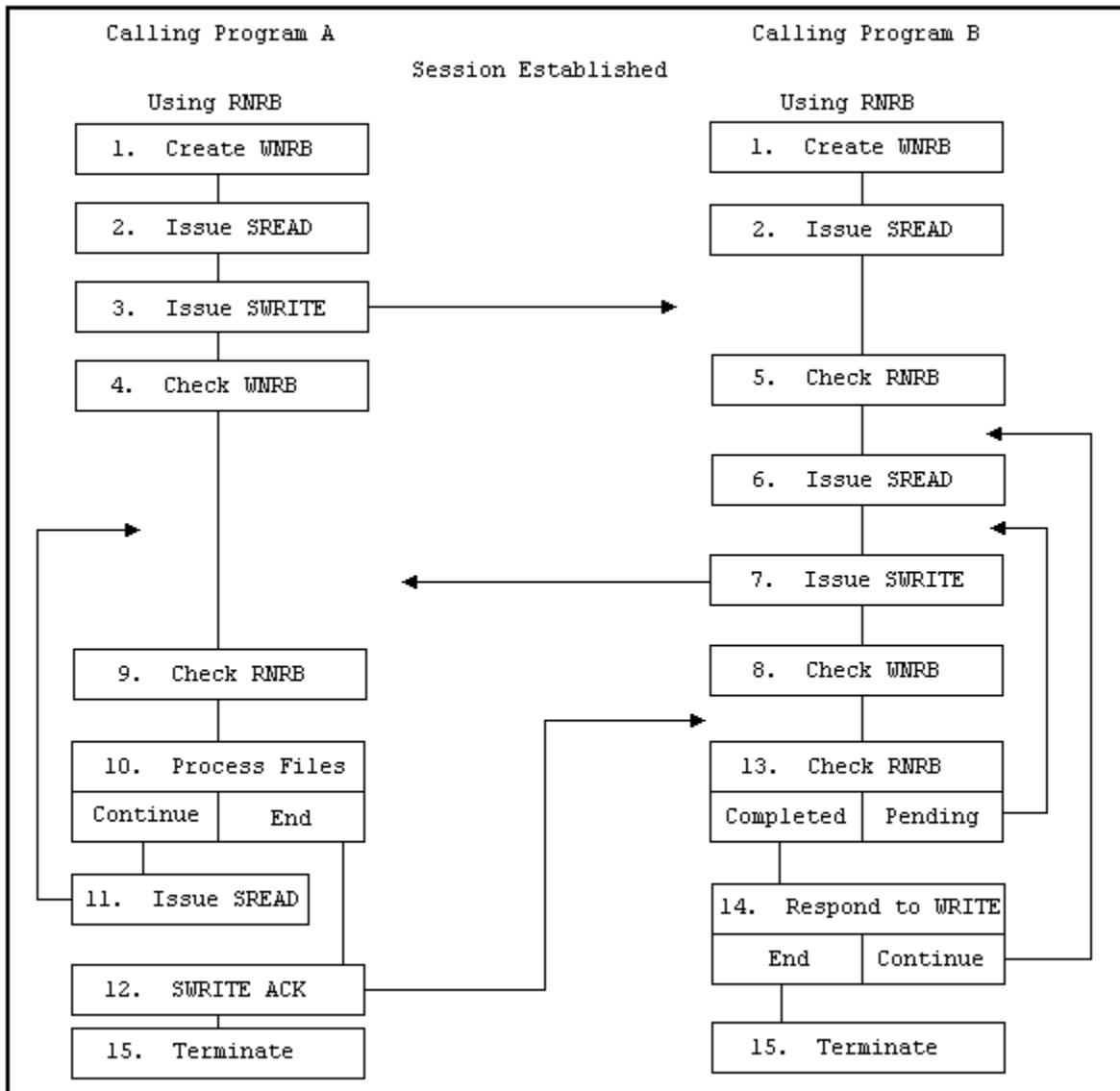
## Concurrent Write and Read Data Transfer

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

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

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





**Figure 8. Concurrent SREAD and SWRITE Requests**

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

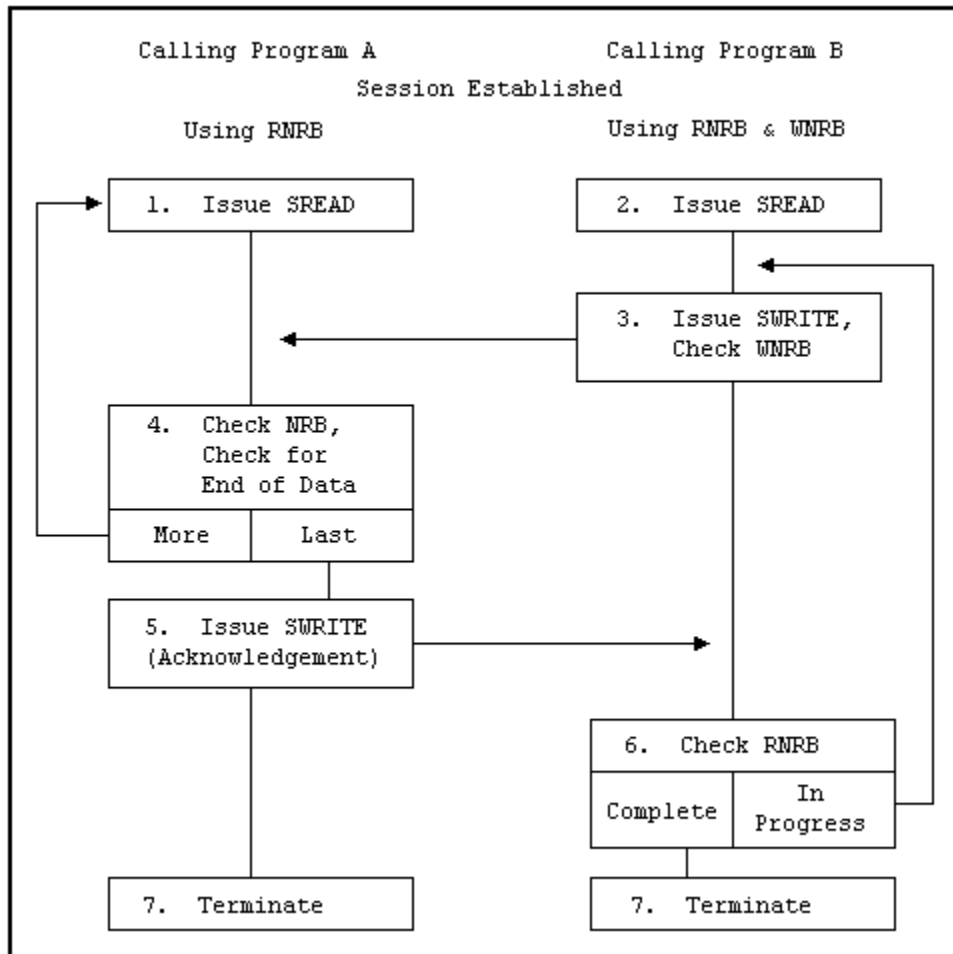
1. After a session has been established, both programs create duplicate NRBs for the SWRITE requests. The NRBs created before the sessions were established are assumed to have been called RNRB, and will be used with SREAD requests. New NRBs called WNRB are duplicates of the RNRB that will be used with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

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

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

## One-Way Data Transfer

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



**Figure 9. One-Way Data Transfer**

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

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

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

7. Program B terminates the session.

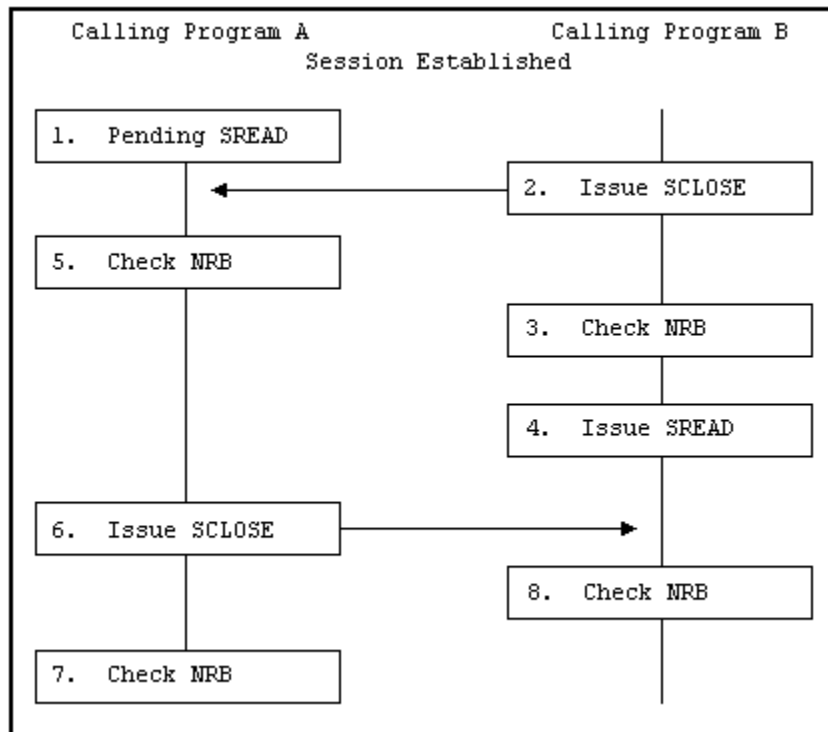
In the previous example, SWAIT requests may be used freely by program A, but should generally be used only after SWRITE requests by program B. An SWAIT could be issued by program B to wait on the RNRB after all data has been written.

## Terminating A Session

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

### Normal Termination

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



**Figure 10. Normal Session Termination**

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

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

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

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

## Abnormal Session Termination

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

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

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

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

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

## Programming Notes

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

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

## Handling Multiple Connections

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

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

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

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

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

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

## Service WAIT Options

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

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

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

The following points apply to waiting:

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

## Satellite Communication

The standard NetEx/IP requests may be used when satellite communication facilities are a part of the network. NetEx/IP was designed and implemented with the capability to recover from errors and lost messages using protocols which make the solving of these problems transparent to the user.

**IMPORTANT:** Because of the long propagation delay inherent in the satellite subsystem (approximately 450ms round trip), the communications channel must be kept "full" of data. This is done by using concurrent SREADs and SWRITEs which transmit data in large amounts before having the writing application stop to

wait for an acknowledgement from the reading application. In this way, data is transmitted rapidly and the propagation delay has less effect on performance. (This technique requires a large buffer area.)

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

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

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

## NetEx/IP Error Recovery Procedures

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

### Error Codes

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

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

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

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

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

### Common Error Recovery Procedures

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

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

## Code Conversion

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



# Chapter 4: NetEx Request Block

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

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

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 Usage

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

- Before initiating a connection, the user must clear the NRB, including the OS dependent portion, to zeroes. When the connection is initiated, the user places whichever non-default values are needed in the user part of the NRB, and invokes NetEx/IP service. Once the connection is initiated, the user must not change the OS dependent part of the NRB between calls to NetEx/IP.
- If the calling program is using the connection in a full duplex manner, the user will need to make a copy of the NRB to produce a “read NRB” and a “write NRB.” This copying operation is the copying of all 40 words of the NRB to another area, at a time when the NRB being copied is not active. If a second request for the same connection is issued from the copied NRB, the user interface must detect the condition and handle the new part of the connection accordingly.
- Many NRB values are specified in addressable units. An addressable unit is the amount of information contained in one memory location for that machine. For example, a Unisys Computer Systems 2200 is 36 bits, IBM systems are generally eight bits, etc.
- For multi-threaded implementations, if a session is to be shared between threads, you must insure that each thread has its own copy(ies) of the NRB made while no NetEx/IP calls for that NRB are active. This rule may be broken only if you ‘gate’ the use of the NRB.

## NRB Fields

Figure 11 shows the fields in the NRB. Most NRB fields are one long word (32 bits) or two words if the length is 16 bits. The NRB contains 40 fields.

Many of the NRB fields are, or could be, updated by either program or NetEx/IP with every request. However, the fields NRBCCLASS, NRBMAXRT, NRBBLKI, NRBBLKO, NRBRVS, NRBOFFER, and NRBHOST are associated with the session negotiation process. Information in these fields is updated by NetEx/IP as their values change. These fields are initially specified during the OFFER and CONNECT requests. 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 Assembler programs using the names shown in Figure 11 if the NTXNRB macro is used. Otherwise, the NRB fields must be referenced using the index numbers shown on the left side of Figure 11.

1	nrbstat	NetEx/IP request status returned to user
2	nrbind	Data type indication from OFFER/READ
3	nrbllen	Length of data
4	nrbubit	Unused bit count
5	nrbreq	User request code
6	nrbnref	NetEx/IP reference number identifying connection
7	nrbbufa	Starting address of user's buffer
8	nrbbufl	Length of user's buffer
9	nrbdmode	Datamode for Write request
10	nrbtime	Request timeout in seconds
11	nrbclass	Class of service
12	nrbmaxrt	Maximum data rate permitted
13	nrbblki	Maximum buffer size for input requests
14	nrbblko	Maximum buffer size for output requests
15	nrbprota	Address of Odata
16	nrbprotl	Length of Odata
17	nrbresv1	Reserved
18	nrbresv2	Reserved
19	nrboffer	(Session Level) Offer name (8 bytes)
21	nrbhost	(Session Level) Remote hostname (8 bytes)
23	nrbresv3	Reserved
24	nrbuser	User field: not processed by NetEx/IP
25-40	nrbosd	Reserved (operating system dependent data)

**Figure 11. NetEx Request Block (NRB) Words**

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

## **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, the NRBSTAT is 0. If the request was unsuccessful (NetEx/IP or the service routine detected an error), NRBSTAT contains a binary representation of a decimal error code. The meanings of the error codes are described in Appendix A.

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

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

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

## **NRBIND**

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

The values returned in NRBIND are defined as follows:

**INDCONNECT** (1) – Connect indication

**INDCONFIRM** (2) – Confirm indication

**INDDATA** (3) – Normal data indication

**INDEXPDATA** (4) – Expedited data

**INDCLOSE** (5) – Close indication

**INDDISCONNECT** (6) – Disconnect indication

**INDSTATUS** (7) – Status Indication

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

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

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

## **NRBLEN and NRBUBIT**

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

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

**Note:** Those programs that do not need the NRBUBIT can simply ignore its existence, knowing that simply handling the information specified by NRBLLEN 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, NRBLLEN will be set to zero.

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

## NRBREQ

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

NRBREQ has the following format:



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

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

**1xxx to 7xxx** - Reserved

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

**9xxx to Fxxx** - Reserved

**Service Level** indicates whether the request is a SESSION, TRANSPORT, NETWORK, or DRIVER type of request. 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 (affects 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, N, and D levels)

**xx02** - Confirm (valid for S, T, N, and D levels)

**xx03** - Write (valid for S, T, N, and D levels)

**xx04** - Reserved

**xx05** - Close (valid for S and T levels)

**xx06** - Disconnect (valid for S, T, N, and D levels)

**xx07 to xx80** - Reserved

**xx81** - Offer (valid for S, T, and N levels)

**xx82** - Read

**xx83** - Status

**xx84 to xxFF** - Reserved

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

## NRBNREF

NRBNREF is the 16-bit, internal NetEx/IP identifier that distinguishes this connection from all others maintained by this copy of NetEx/IP. When initial connect or offer requests are made at the Driver level, this field must be filled in by the caller. If issued at the Session level, this value is assigned by NetEx/IP when a connection is established.

## NRBBUFA

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

## NRBBUFL

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

## NRBDMODE

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

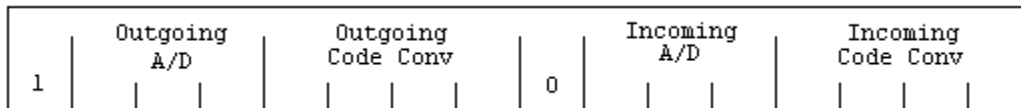
Datamode supports two basic modes: manual and auto datamode.

### Manual Datamode

Manual datamode allows complete specification of the assembly/disassembly and code conversion functions on both output and input. In the manual datamode, the caller has total control over the facilities. The user must determine which assembly/disassembly modes and code conversion tables are meaningful to the two ends involved in the transfer.

The datamode field is always in the “datamode” field of the drive 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.

Manual datamode has the following format:



‘1’ is the high order bit is the manual mode indicator

**Outgoing A/D** indicates the data assembly/disassembly to be performed on data as it goes out onto the network. This information is added to the “transmit data” function code when the user data goes over the network.

**Outgoing Code Conv** indicates the code conversion to be performed on data as it goes out onto the network. This information is added to the “transmit data” function code when the user data goes over the network.

‘0’ in the high order bit of the low order byte is the manual mode indicator.

**Incoming A/D** indicates the data assembly /disassembly to be performed on data as it goes from the network to the receiving program. This information is added to the “input data” function code when the receiving driver gets the message from the network.

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

### Auto Datamode

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

Auto datamode supports three conversion options.

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

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

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

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



‘0’ in the high order bit is the auto mode indicator.

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

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

**Destination Character Set** indicates the conversion option of the data going to the destination program. For example, a conversion from EBCDIC (3) to ASCII(2) would be entered as a hexadecimal value of 0302 or decimal value of 770.

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

The processing rules for auto datamode are as follows:

- 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 each sent over the network as an eight-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 the 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 eight-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 end has hardware 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 "meaningful" on its own host, then it treats the incoming character set as "octet mode" data and provides the user with the data along with an error code in NRBSTAT indicating that the data was "damaged."
- If the destination character set is a six-bit code, code conversion hardware is required on the interface for the six-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).
- 1 Use Version 1 NetEx/IP protocol. This protocol is no longer supported.
- 2 Use Version 2 NetEx/IP protocol. This protocol is used in Release 2 and above NetEx/IP products.
- 4 Use Version 4 NetEx/IP protocol. This protocol is supported by NetEx/IP Versions 6 and above.

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

When an offer or connect completes, the value of this field should contain the protocol version actually negotiated. 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. These levels are not discussed in this manual. 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/IP 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 16-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:** The NRBMAXRT and the throttling concept only directly apply to the transmitting portion of a given connection. The other party in the connection may be working with completely different throttling parameters and the corresponding program will have no direct way of knowing the remote transmitter’s data rate parameters.

On completion of the offer or confirm request, this field will have a nonzero value that contains the maximum throughput that is possible to the connection, based on the user’s original request and the characteristics of the communications 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/IP installation systems programmer must supply two values controlling these buffer sizes. First, the default input and output block sizes to be used if these are not specified (left zero) by the caller. Secondly, the maximum input and output block sizes permitted by the installation.



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

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

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/IP 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 that the values implied using zero or -1 will be used for negotiation of the connection block sizes. The actual size negotiated will be supplied in these fields on completion of the connect of offer.

## NRBPROTA and NRBPROTL

The NRBPROTA and NRBPROTL are connection negotiation parameters that permit the application to provide Odata to the called layer of NetEx/IP. NRBPROTA specifies the address of the buffer containing the Odata, and NRBPROTL specifies the number of octets of Odata in that buffer.

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

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

The maximum amount of Odata that may be sent is limited. The maximum is installation dependent and may typically be 256 bytes or less. Each version of NetEx/IP will have a generated maximum on the number of bytes of Odata that it is prepared to accept in incoming messages. In the Session level, the maximum amount of Odata that may be sent or received will be the minimum of the Odata sizes generated on each host.

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

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

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

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

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/IP enhancements.

## **NRBOFFER**

This field is a session negotiation parameter that consists of two 16-bit fields. 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 meaningful in ASCII should be used during the name matching process.

## **NRBHOST**

This field is a session negotiation parameter that consists of two 16-bit fields. 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.

## **NRBRESV3**

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

## **NRBUSER**

This field is reserved for users. Most typically it will be used to pass information to the User Exits that exist in host based NetEx/IP implementations. NetEx/IP will not process this field in any way.

## **NRBOSD**

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

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

## **Creating an NRB**

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

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

## **Duplicating an NRB**

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

To duplicate an NRB, wait until the initial CONNECT or OFFER has completed successfully, then copy the entire "working" NRB (up to and including the NRBOSD field) to a blank NRB at a different location. The

second NRB can now be used for NetEx/IP requests. The NRB should only be duplicated when it is not in use; that is, when NRBSTAT is 0.



# Chapter 5: C High Level Interface

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

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

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

## C NETEX Request Blocks

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

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

#define	INDCONNECT	1
#define	INDCONFIRM	2
#define	INDDATA	3
#define	INDEXPDATA	4
#define	INDCLOSE	5
#define	INDDISCONNECT	6
#define	INDSTATUS	7

Using these types the fields are as shown in the following table:

TYPE	FIELD	Function
long	nrbstat	Status or error code
long	nrbind	Indication
long	nrbllen	Length of data
long	nrbubit	Unused bit count
long	nrbreq	Request code
long	nrbref	Reference number
char	nrbbufa	Pointer to user's buffer

<b>TYPE</b>	<b>FIELD</b>	<b>Function</b>
long	nrbbuf1	Length of user's buffer
long	nrbdmode	Data mode of write request
long	nrbtime	Request timeout in seconds
long	nrbclass	Class of service
long	nrbmaxrt	Maximum transmission rate
long	nrbblki	Input buffer size
long	nrbblko	Output buffer size
char*	nrbbprot1	Address of Odata
long	nrbprot1	Length of Odata
long	nrbresv1	PAM
long	nrbrrref	Remote reference number
char	nrboffer[8]	Offer name
char	nrbhost[8]	Remote host name
long	nrbresv3	Reserved
long	nrbuser	Reserved for users
long	nrbosd[16]	OS Dependent

## SOFFR C Function

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

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

### SOFFR Function Format

The SOFFR function has the following format:

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

### SOFFR Parameters

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

#### soffr

#### soffrw

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

#### nrb

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

#### buffer

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

#### length

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

#### timeout

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

#### pname

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

## SOFFER Entry Parameters

The following NRB fields are used by SOFFR on entry.

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

## SOFFR Results

The following NRB fields are updated when SOFFR completes.

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



# SCONN C Function

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

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

## SCONN Function Format

The SCONN function has the following format:

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

## SCONN Parameters

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

### sconn sconnw

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

### nrb

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

### buffer

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

### length

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

### datamd

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

### pname

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

## hname

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

## SCONN Entry Parameters

The following NRB fields are used by SCONN on entry.

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

## SCONN Results

The following NRB fields are updated when SCONN completes.

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

## SCONF C Function

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

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

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

### SCONF Function Format

The SCONF function has the following format:

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

### SCONF Parameters

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

#### **sconf** **sconfw**

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

#### **nrb**

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

#### **buffer**

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

#### **length**

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

#### **datamd**

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

### SCONF Entry Parameters

The following NRB fields are used by SCONF on entry.

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

## **SCONF Results**

The following NRB fields are updated when SCONF completes

<b>NRBSTAT</b>	Success/failure code
<b>NRBIND</b>	Set to zero

# SREAD C Function

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

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

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

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

## SREAD Function Format

The SREAD function has the following format:

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

## SREAD Parameters

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

### sread sreadw

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

### nrb

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

### buffer

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

### length

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

### timeout

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

## SREAD Entry Parameters

The following NRB Fields are used by SREAD on entry.

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

## SREAD Results

The following NRB fields are updated when SREAD completes.

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

# SWRIT C Function

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

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

## SWRIT Function Format

The SWRIT function has the following format:

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

## SWRIT Parameters

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

**swrite**  
**swritw**

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

**nrb**

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

**buffer**

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

**length**

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

**datamd**

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

## SWRIT Entry Parameters

The following NRB fields are used by SWRIT on entry.

<b>NRBBUFA</b>	Address of outgoing Pdata
<b>NRBLEN</b>	Length of outgoing Pdata
<b>NRBUBIT</b>	Pdata unused bit count
<b>NRBDMODE</b>	Datamode of Pdata
<b>NRBPROTA</b>	Address of outgoing Odata

**NRBPROTL**            Length of outgoing Odata

## **SWRIT Results**

The following NRB fields are updated when SWRIT completes.

**NRBSTAT**            Success/failure code

**NRBIND**            Set to zero



## SCLOS C Function

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

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

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

### SCLOS Function Format

The SCLOS function has the following format:

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

### SCLOS Parameters

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

**sclos**  
**sclosw**

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

**nrb**

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

**buffer**

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

**length**

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

**datamd**

This required parameter is the datamode to be used to send the connect data to the corresponding application. The data type of *datamd* should be INT. Refer to NRBDMODE in chapter 4 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

<b>NRBLEN</b>	Length of outgoing Pdata
<b>NRBUBIT</b>	Pdata unused bit count
<b>NRBDMODE</b>	Datamode of Pdata
<b>NRBPROTA</b>	Address of outgoing Odata
<b>NRBPROTL</b>	Length of outgoing Odata

## **SCLOS Results**

The following NRB fields are updated when SCLOS completes

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

## SWAIT C Function

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

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

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

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

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

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

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

## C SWAIT Examples

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

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

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

**Figure 12. SWAIT(0) Example**

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

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

Figure 13. SWAIT(-1) Example

## SWAIT Function Format

The SWAIT function has the following format:

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

## SWAIT Parameters

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

### swait

This is the verb for this function.

### nrbnum

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

### nrb

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

# SDISC C Function

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

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

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

## SDISC Function Format

The SDISC function has the following format

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

## SDISC Parameters

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

### sdisc sdiscw

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

### nrb

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

### buffer

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

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

### length

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

### datamd

This required parameter is the datamode to be used to send the disconnect data to the corresponding application. The data type of *datamd* should be INT. Refer to NRBDMODE in chapter 4 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.

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

## SDISC Results

The following NRB fields are updated when SDISC completes.

<b>NRBSTAT</b>	Success/failure code
<b>NRBIND</b>	Set to zero

## C Program Examples

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

To compile these programs use:

```
cc -o nsef001 nsef001.c -lntxc
cc -o nsef002 nsef002.c -lntxc
```

To run the programs:

```
nsef001 [datamode]
nsef002 hostname [datamode]
```

### datamode

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

To run intra-host:

```
nsef001 &
nsef002 localhostname
```

Figure 14 follows.

```
/*
 *This is the offering side of a connection, the basic sequence is
 * offer with data verification
 * confirm
 * read and write to remote with data verification
 * read disconnect
 * the session is terminated if there are any errors
 */

#include<netex.h>
#define SUCCESS 0

nrb n;
short buf[500], verf[500]
int step, i, mode;
int bytes=2;
char *job[]={"offer", "connect", "confirm", "read", "write", "disc", "wait"};
main(argc,argv) /* nsef001 [datamode] */
int argc;
char *argv[];
{
    memset(&n, '0', sizeof(n));
    memset(buf, '\0', sizeof(buf));
    step=0;
    printf("this is the start of nsef001\n");
    if(argc<2)
        mode=0;
    else
        sscanf(argv[1], "%x", &mode);
    verf[0] =1234;

    /*offer, check for success, verify data*/
```



```

soffrw(&n,buf,4,300,"NSEF0011");
if(n.nrbstat !=SUCCESS || n.nrbind !=INDCONNECT)
    terminate(0);
if(buf[0] !=verf[0])
    terminate(1);
buf[0] =1234;
buf[1] =5678;

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

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

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

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

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

/*read the disconnect */

```

```

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

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

/*
 * this is the connecting side of the connection, the basic sequence is
 * connect to remote
 * read confirm with data verification
 * write to and read from remote host with data verification
 * disconnect
 */
#include<netex.h>
#define SUCCESS 0

nrb n;
short buf[500],verf[500];
int step,i,mode;
int bytes =2;
char *job[]={ "offer","connect","confirm","read","write","disc","wait"};
char host[9]

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

```

```

else {
    printf(" nsef002: wrong arg count\n");
    exit(1);
}
memset(host, ' ', sizeof(host)-1);
memcpy(host, argv[1], strlen(argv[1]));
verf[0] =1234;
verf[1] = 5678;
buf[0] =1234;

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

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

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

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

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

```

```

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

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

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

```

**Figure 14. C Program Example**

# Chapter 6: TNP

## Overview

The TNP feature serves as a NetEx/IP “proxy”, which can be used by other hosts for which a host-based NetEx/IP is not available. A NetEx/IP Requester, residing on an HP NonStop, OpenVMS, Bull, or Stratus server, works with BFX, USER-Access, eFT, PFX (or other NetEx/IP applications running on the Requester server), and reads and writes the application’s NetEx/IP requests over a TCP/IP connection to TNP on Linux, which then passes the request to Linux NetEx/IP, on behalf of the Requester’s NetEx/IP application. In effect, TNP serves as a “proxy” NetEx/IP application for the Requester’s NetEx/IP applications.

Prior to UNIX NetEx/IP with TNP, Requester applications used a NESiGate-LO device to provide NetEx/IP services. UNIX NetEx/IP with TNP eliminates the need for NESiGate-LO. TNP is not supported on the Windows NetEx/IP product.

TNP is a separately licensed feature of UNIX NetEx/IP. If enabled, a TNP NetEx/IP application is started during NetEx/IP initialization, and is ready to accept connections from NetEx/IP Requester applications running on different servers. When a NetEx/IP request comes in from a Requester, TNP becomes the local NetEx/IP application, acting on behalf of the Requester. TNP uses the NetEx/IP API’s to communicate with the local NetEx/IP, just like any other local NetEx/IP applicaion. The Requester application is able to establish NetEx/IP sessions with other remote NetEx/IP hosts on the network, as well as with UNIX NetEx/IP. Connections established between Requester applications and UNIX NetEx/IP effectively appear as NetEx/IP intrahost connections.

The NetEx/IP license key contains a session limit count, which places an upper limit on the number of concurrent TNP NetEx connections.

When running Netex TNP applications from Netex Requesters, it is recommended that a NetEx/IP Requester host entry be added to each of the host-based NetEx/IP NCT’s, and that the multihost feature be enabled in the UNIX NetEx/IP that is supporting TNP.

This can be done by specifying MULTIHOST ON in the ntx\_default initialization file, or by issuing the SET MULTIHOST ON NetEx/IP command. To ensure this setting is persistent across NetEx/IP restarts, the MULTIHOST statement should be added to the ntx\_default file. Refer to the “Adding NetEx/IP Requester Hosts to the NCT” section on page 64 for more information on the NCT requirement.

shows a display of a TNP requester BFXJS application, submitted on host TNPFLASH (a NetEx/IP Requester system). Unique Requester user names are assigned by TNP. They are prefixed with ‘TNP’, followed by 5 numerics.

Host Sref	UNIXT User	Active Sessions Pid	State	Name	Host	RNref	Msg In	Msg Out	Tid
1	NTXOPER	0	OFFERED	NTXOPER	UNIXT				0000
4	root	3416	OFFERED	TNPOFFER	UNIXT				0000
5	TNP00001	22498	OFFERED	BFXJS	TNPFLASH				57e2

Figure 15. Output display of an OFFERED TNP job

Figure 16 shows a display of several active TNP connections. Since these are all connections between TNPFLASH and UNIXT, these are effectively all intrahost NetEx/IP connections on UNIXXT, but transferring files between the NetEx/IP Requester host and UNIXT.

Host UNIXT		Active Sessions								
Sref	User	Pid	State	Name	Host	RNref	Msg In	Msg Out	Tid	
1	NTXOPER		OFFERED	NTXOPER	UNIXT				0000	
4	TNP		OFFERED	TNPOFFER	UNIXT				0000	
5	BFXJST		OFFERED	BFXJST	UNIXT				0000	
87	TNP00002	3550	CONNECTD	NTXGENDC	UNIXT		4901	0	0dde	
88	NTXGENDC	3600	CONNECTD	TNP00002	UNIXT	126	1	4911	0e10	
94	BFXTNPFL	22480	CONNECTD	TNP00001	UNIXT		2	121	57d0	
95	TNP00001		OFFERED	BFXJS	TNPFLASH				0000	
96	TNP00001	12345	CONNECTD	BFXTNPFL	UNIXT	138	110	1	3039	

Figure 16. Output display of connected TNP jobs

## TNP Configuration

### NetEx/IP License Key

The TNP software is a feature which must be enabled in the NetEx/IP license key.

### TNP PROGRAM

During the UNIX NetEx/IP installation, the TNP program is installed and the default tnp.cfg file is put in /etc/opt/netex. However, this file can also be edited and changed, if needed, anytime after the UNIX NetEx/I installation. If TNP is enabled (via software key), then the TNP program gets started during NetEx/IP initialization. The TNP program issues the TNPOFFER, then waits for work requests from remote NetEx/IP Requesters.

## tnp.cfg

tnp.cfg is the TNP configuration file, and is shown in Figure 17. The default tnp.cfg file is installed during installation.. However, this file can also be edited and changed, if needed, anytime after the HxxOIP installation

PORT keyword specifies the port number used with NetEx/IP Requester hosts, and must be the same as the port specified by the ‘TCP’ parameter in the NetEx/IP Requester host configuration file.

CMDPORT number specifies a command port used for internal TNP connections.

OFFRNM is the name of the TNP NetEx offer. This is only used in the display NetEx sessions.

MSGSYSLOG specifies where TNP messages will be output. A value of 1 means output to the tnp.log. A value of 2 means output to the system log. A value of 3 means output to both logs. A value of 0 means no syslog and minimal log output.

MSGSYSLOGFAC specifies how syslog messages will be output. A default syslog facility code of “local3” will be used unless overridden. The valid facility codes must be used if overridden. It is recommended only the local facility codes be used. Check with your system administrator for more information.

The syslog priority for TNP is INFO, Priority 6.

```
#
#   TNP CONFIGURATION
#   =====
#
#   KEYWORD      VALUE                                     DEFAULT
#   -----
#   TNP          (NONE)
#   PORT         PORT NUMBER USED WITH                 5001
#               NetEx/IP REQUESTER(S)
#   CMDPORT     PORT NUMBER USED FOR                   7002
#               COMMAND PROCESSING
#   OFFRNM      OFFER NAME                             TNPOFFER
#
TNP port 5001 cmdport 7002 offrnm TNPOFFER
```

Figure 17. Requester host configuration file

## tnpop

tnpop is a minimal operator interface to dynamically set the TNP configuration parameters which can be set persistantly in the tnp.cfg file. The following commands are supported:

**help** – display the list of possible tnpop commands

**debug (on|on2|off)** – set the level of debugging messages to the tnplog (on or off ) on2=on

**tv** – display TNP’s internal version

**logcmds (on|off)** – turning on/off the logging of tnpop commands to the tnplog (and output of valid commands)

**histlist** – displays a running total of TNP session count per requestor

## Adding NetEx/IP Requester Hosts to the NCT

For TNP configurations, an additional HOST entry may optionally be defined in the NCT to identify each NetEx/IP Requester host. This NCT change should be made to all of the host-based NetEx/IP NCT's. The name on the HOST statement identifying the NetEx/IP Requester host must be the same name as specified by the 'local' keyword in the NetEx/IP Requester configuration file. Additionally, the following NetEx/IP statement must be added to the ntx\_default initialization file:

```
MULTIH ON
```

The recommended configuration is that the NetEx/IP Requester hosts should be added to the NCT file. Doing this will allow greater control over how connections get established in cases where duplicate NetEx/IP application OFFER names exist on multiple hosts. Without having specific entries in the NCT per Requester host, and not specifying MULTIH ON, it can be somewhat unpredictable how connections to duplicate OFFER names get established, since all applications would appear as Linux intrahost connections.

An additional NCT requirement for NetEx/IP Requester hosts is that the ADAPTER defined for the NetEx/IP Requester HOST must be the same as an Adapter defined for the local Hxx0IP host.

An example of an HXX0IP NCT with a NetEx/IP Requester host is shown in Figure 18.

```
*
** Local HOST LINUXT
*
LINUXT  HOST      TYPE=LINUX  MODEL=LINUX  PROTOCOL=2
*
        ADAPTER MODEL=N400  NETADDR=01
        T0=IP1
        SMGDREF=00  PORT=0
*
** NetEx/IP Requester host (uses TCP/IP as NetEx/IP intrahost path between
** Linux NetEx/IP and NetEx/IP Requester host)
*
TNPFLASH HOST      TYPE=LINUX  MODEL=LINUX
        PROTOCOL=2
        ADAPTER MODEL=N400  NETADDR=01
        T0=IP1
        SMGDREF=00  PORT=0
*
```

**Figure 18. Sample NCT with NetEx/IP Requester host**



## Requester Configuration

The NetEx/IP Requester host also has a configuration file. Refer to the appropriate NetEx/IP Requester manual for a description of the configuration file. It contains the IP address and port number used when communicating with HXXOIP TNP.

The "local" keyword specifies a name for the local client host. If MULTIHOST is set (recommended), this name is used by the TNP NETEX as the NETEX hostname for sessions offered by the local client host. The name specified should be added to the NetEx/IP Network Configuration Table (NCT) on all of the host-based NetEx/IP platforms. (Refer to "Adding NetEx/IP Requester Hosts to the NCT" on page 64).

The "netex" keyword identifies the TNP host. The name following the "netex" keyword identifies a NetEx/IP host that supports the TNP feature. The third value specifies the IP address of the NetEx/IP TNP host. The 'tcp' keyword specifies the port number used by the HXXOIP TNP component, and must be the same as the port specified in the HXXOIP TNP configuration file. If the port number is not specified, the default port number (5001) is used.

An example of a NetEx/IP Requester configuration file is shown in Figure 19.

```
#
# Sample NetEx/IP Requester configuration
#
# The "local" keyword specifies a name for the local client host.
#
local tnpflash
#
# The "netex" keyword identifies the TNP host, along with the IP address and
# port number used by TNP.
#
netex LINUXT 10.1.5.156 tcp 5001
```

**Figure 19. Sample NetEx/IP Requester configuration**



# Chapter 7: Installation

Please refer to the appropriate appendix for installation and OS specific information for your operating system:

- Appendix D: H800IP Linux Installation
- Appendix E: H620IP AIX Installation
- Appendix F: H820IP Linux for System z Installation



# Chapter 8: Operator Interface

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

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

# Executing Commands

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

Any characters following the command and parameter entries will be ignored

## Command Line Mode

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

```
ntxoper operator_command parameters
```

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

```
ntxoper display host
```

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

```
ntxoper : host_name operator_command parameters
```

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

## Interactive Mode

When you execute operator commands interactively, you first start an operator interface session using the NTXOPER command. During an NTXOPER session, all valid operator commands are available. you can also use these additional commands:

Interactive Operator Command	Description
. (period)	Repeats the previous command
rmt hostname operator_command	Executes the specified operator command using the remote NTXOPER interface.
crmt hostname operator_command	Executes the specified operator command using the remote NTXCONS interface.
QUIT	Terminates NTXOPER. You can also terminate NTXOPER by entering EXIT, BYE, or STOP.

## Enabling the Remote Operator Service

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

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

```
ntxoper set ntxoper on
```

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

```
ntxoper set ropclass a
```

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

```
ntxoper set ropclass g
```

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

# Operator Command Descriptions

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

<b>Operator Command</b>	<b>Description</b>
CLEAR LOG	Clears the NETEX log file
CLEAR IPROUTE	Clear IP routing table entries
DISPLAY DRAINED	Displays drained adapters
DISPLAY HOST	Displays host(s) defined to NETEX
DISPLAY IPROUTE	Displays routing table entries
DISPLAY KEY	Displays the current license key
DISPLAY LOG	Displays the last 100 lines of the NetEx/IP log
DISPLAY MEMORY	Displays the current memory allocations for various internal control blocks and data buffers. (Same as the DISPLAY MEMORY command.)
DISPLAY NETWORK	Lists network connections currently pending or in progress
DISPLAY PARMS	Displays parameter values controlled by SET commands
DISPLAY SESSION	Lists sessions currently pending or in progress
DISPLAY TRANSPORT	Lists the current state of transport connections
DISPLAY USAGE	(See DISPLAY MEMORY command)
DISPLAY VERSION	Displays the current version level of the NetEx/IP component.
DRAIN ADAPTER	Prevents new connections from using the adapter
DRAIN HOST	Prevents new connections with a specific host
DRAIN NETEX	Prevents any new connections from starting
HALT SREF	Immediately stops a NetEx session
HELP / lhelp	Provide NetEx/IP help information.
KILL NETEX	Immediately stops all NETEX resources
LOAD KEY	Loads the current license key
LOAD NCT	Transfers a pamfile created by the Configuration Manager to NETEX
SET CONTIME	Specifies the connect attempt timeout value
SET DBGDATA	Specifies number of data bytes to trace
SET DBGMSG	Enable/disable NETEX message tracing
SET DBGREQ	Enable/disable user request tracing
SET DBGRET	Enable/disable user response tracing



<b>Operator Command</b>	<b>Description</b>
SET DEADTIME	Specifies time to wait until disconnecting a connection due to lack of traffic
SET DEFBI	Specifies the default maximum input buffer size for sessions
SET DEFBO	Specifies the default maximum output buffer size for sessions
SET IDLETIME	Specifies time between idle messages to verify the continued existence of the other end of the connection
SET IPROUTE	Set an IP routing table address entry
SET MAXBI	Specifies the maximum input buffer size a user may request on a SCONN or SOFFR
SET MAXBO	Specifies the maximum output buffer size a user may request on a SCONN or SOFFR
SET MAXKBS	Specifies the maximum rate at which NetEx/IP will deliver data to the network for each network connection.
SET MSGLVL	Specifies the level of messages to display
SET MULTIHOST	Set multihost option on or off
SET NTXOPER	Enables and disables the remote operator service
SET PREFPROT	Specifies the default preferred protocol type to use when connecting to hosts that support multiple NetEx/IP protocol types.
SET READTIME	Time that NETEX will retain incoming data waiting for a user read request
SET ROPCLASS	Class of operator commands that the remote operator may issue
SET SESMAXIMUM	Number of concurrent session offers and active connections allowed
SET WDOGINT	Time between NETEX internal checks for timed events
START NETEX	Reverses the effect of DRAIN NETEX
START HOST	Reverses the effect of DRAIN HOST
SWLOG	Saves the current 'ntxlog' file as 'ntxlog.n' (where <i>n</i> is an integer greater than or equal to 1), and starts using a new 'ntxlog' file. This command is not available in all implementations.

**Figure 20. NETEX Operator Commands**

## **CLEAR LOG**

### **Description**

Clears the NETEX log file while NETEX is running.

### **Format**

Clear LOG

# CLEAR IPRROUTE

## Description

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

## Format

```
Clear IProute uuss | ALL
```

## Parameters

### uuss

The NCT defined NETADDR and SMGDREF of a host adapter.

### ALL

Clear all table entries

## **DISPLAY DRAINED**

### **Description**

The operator may display a list of drained adapters.

### **Format**

Display Drained

# DISPLAY HOST

## Description

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

## Format

Display Host [hostname]

## Parameters

### hostname

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

## Display Examples

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

13:12:41 Host ULTRA5H						Current Routes					
DestHost	Pr	Rts	State	Fl	PmOf	DestHost	Pr	Rts	State	Fl	PmOf
NTXLCL	2	3				NTXLCL00	2	1			
DALE	2	4				TANDEM	2	4			
TANDEMSH	2	4				DXU20	2	2			
MINGH	2	4				FLASHH	2	4			
DXUB3	2	2				HP	2	5			
STRATUS	2	5				RIOS4	2	4			
SOLSRVR	2	5				ULTRA5	2	3			
UNISYS	2	7				ZARKHOV	2	4			
ZARKGWH	2	1				ZARKHOVE	2	1			
UNISYSE	2	2				UNISYSH	2	2			
ULTRA5H	2	1				SOLGWH	2	1			
SOLSRVRF	2	1				SOLSRVRH	2	1			
RIOS4F	2	1				RIOS4H	2	1			
STRATUSH	2	1				AIX1GWE	2	1			
AIX1F	2	1				AIX1H	2	1			
OS390E2	2	1				OS390H	2	2			
NETFIN5	2	1				NETFIN3	2	2			
HPE	2	1				HPF	2	1			
HPH	2	1				DXUB3GWH	2	1			
DXU20H	2	1									

Figure 21. Display Hosts Command Output, No HostName specified.

The columns in the display are as follows:

### DestHost

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

### Pr

Type of protocol supported by the destination host, as defined in the NCT data file HOST statement.

### Rts

The number of routes defined by the Configuration Manager to reach the destination host.

**State**

If a host is currently drained, DRAINED appears for that host.

**Fl**

The tplflags for that host (currently could be x'02' (ALTFIRST)

**PmOf**

The offset for the first pam to be used (can be non-zero only if ALTFIRST used for the host)

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

```
13:18:59      Host ULTRA5H      Routes to os390h
pam header - len= 14 segsize= fffb
              maxrate=    0 delay=    0
pam entries --
  pam entry  1 -> 6  1 80 88 db  0
  pam entry  2 -> 6  1 40 88 cb 40
-----
pam header - len= 14 segsize= fffb
              maxrate=    0 delay=    0
pam entries --
  pam entry  1 -> 6  1 80 88 db  0
  pam entry  2 -> 6  1 40 88 20 80
-----
```

**Figure 22. Display Host Command, HostName specified**

The fields in the display are as follows:

**Host**

This field shows the NETEX name of the local host.

**Routes to**

This field shows the NETEX name of the destination host.

**len**

This field shows the length of the PAM for this route.

**segsize**

This field shows the maximum segment size supported on this route.

**maxrate**

This field shows the maximum rate of transmission on this route, specified in 1000s of bits per second.

**delay**

This field shows the amount of time to wait between successive transmissions on this route. Time is specified in milliseconds.

**pam entry**

This field shows the PAM route component entry for each adapter in the route (length, type code, flag, trunk mask, dref [two bytes]).

# DISPLAY IPROUTE

## Description

Display all or a selected IP routing table entry.

## Format

```
Display IProute [ uuss | ALL ]
```

## Parameters

### uuss

The NCT defined NETADDR and SMGDREF of a host adapter.

### ALL

Display all table entries. All is assumed if no uuss is given.

## Display Examples

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

11:42:34 Host STEVE1			Current GNA to IP Mapping Table:		
GNA	IP Adr	Source	GNA	IP Adr	Source
-----	-----	-----	-----	-----	-----
00002000	10.1.2.25	Local	00002080	10.1.2.25	Local
00002480	5.5.5.5	Oper	00002400	10.1.2.25	Local
00003401	0.0.0.0	Unknown	00005181	6.6.6.6	Oper
00005101	0.0.0.0	Unknown	00005201	0.0.0.0	Unknown
00005501	0.0.0.0	Unknown	00005701	10.1.5.26	Local
00005801	0.0.0.0	Unknown	00005901	0.0.0.0	Unknown
00006100	10.1.2.27	Local	0000a101	10.1.100.5	Local
0000a201	192.168.100.4	Local	0000a401	10.1.2.25	Local
0000a501	10.1.5.26	Local	0000a600	10.1.2.27	Local
0000a801	10.1.2.29	Local	0000a901	10.1.5.28	Local
0000ab01	10.1.5.17	Local	0000ac01	10.1.2.10	Local
0000ad01	10.1.2.5	Local	0000ae01	10.1.2.3	Local
0000af01	10.1.2.1	Local	0000b3b0	0.0.0.0	Unknown
0000b5c0	10.1.2.25	Local	0000b801	10.1.5.38	Local
0000b901	10.1.5.39	Local	0000ba01	10.1.2.24	Local
0000c6c0	10.1.5.27	Local	0000cb40	10.1.2.25	Local
0000d805	0.0.0.0	Unknown	0000d905	10.1.2.25	Local

**Figure 23. Display IP Route Command**

The columns in the display are as follows:

### GNA

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

### IP Adr

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

**Source**

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



## DISPLAY KEY

### Description

Displays the NETEX license key

### Format

Display Key

### Display Examples

The following figure shows a sample DISPLAY KEY:

```
15:08:05          Host BUDDY3          License Key
ctlcap=44(IP TNP), ctlicflag=00(), ctlnpsmax= 0

Key = CKFZ-4AAA-LIAE-IAMS-OVZJ-F2JS-GBPC-V2FA
Protocol = 44(IP TNP)
Not Operational Date = 20121230
Expiration Date = 20121001
Tnp SesMax = unlimited
```

**Figure 24. Display Key Command**

## **DISPLAY LOG**

### **Description**

Displays the last 100 lines from the NetEx/IP log file.

### **Format**

Display Log

# DISPLAY MEMORY

## Description

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

## Format

Display MEMory

## Display Examples

The following figure shows a sample DISPLAY MEMORY command output:

```
NtxOper> display memory

14:23:42      Host NETFIN10      Memory
ctlbufs=      6 ctlmbufs=      0 ctlnrbs=      22 ctlpam=      17
ctlmsg=      6 ctlodata=      15 ctlquehead=    15
ctlquesub=    2 ctlquetub=      3 ctlquenub=      3 ctlquedcb=      1
ctlquebcb=    6 ctlquendb=      0 ctlquetdb=      3
```

Figure 25. Display Memory Command

### ctlbufs

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

### ctlmbufs

# of dynamically allocated buffers (number of mallocs)

### ctlnrbs

# of nrb control blocks currently allocated

### ctlpam

# of pams currently allocated for active sessions

### ctlmsg

# of message proper's currently allocated

### ctlodata

# of odata buffers currently allocated

### ctlquehead

# of queue headers currently allocated

### ctlquesub

# of session user blocks currently allocated

### ctlquetub

# of transport user blocks currently allocated

### ctlquenub

# of network user blocks currently allocated

**ctlquedcb**

# of device blocks currently allocated

**ctlquecb**

# of data buffers currently allocated

**ctlquendb**

# of network data control *blocks currently allocated*

**ctlquetdb**

# of transport data control blocks currently allocated

# DISPLAY NETWORK

## Description

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

## Format

Display Network [nref]

## Parameters

### nref

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

## Display Examples

The following shows a sample DISPLAY NETWORK command output:

13:29:52	Host	ULTRA5H	Active Network Connections		
Nref	User	Tref	State	RNref	Prot Type
-----	-----	-----	-----	-----	-----
143	PFXU2200	162	DATA	31	2
145	NTXOPER	164	DATA	60	2
-1	SESSMGR2	165	OFFERED		2

Figure 26 Display Network Command

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

09:56:04	Host	ULTRA5H	Nref	65	
Name=	root	Pid=	21587008	State=data	
Writes=	11	Reads=	10	Nubblki= 32768	Nubblko= 32768
Readto=	80				
Physical Address Map (PAM)					
pam header --					
	len=14	segsz=ffff	maxrate=	0	delay= 0
pam entries --					
	pam entry	1->	6	1	80 88 db 0
	pam entry	2->	6	1	40 88 df 1
end of pam					

Figure 27. Display Network Command, NREF specified

The following describes the fields in the display:

### Nref

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

### Name

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

## State

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

<b>closing</b>	A close has been issued by the user. No additional data may be sent, but additional data may be received.
<b>confirm</b>	CONNECT message received, waiting for the confirm call.
<b>data</b>	Connection completed and user may exchange data.
<b>connect</b>	Connect request issued by user, waiting for confirm.
<b>disconnect</b>	Disconnect detected, but not yet complete.
<b>offered</b>	Offer has been issued by user, waiting for connect.
<b>smgr conn</b>	A user is in the process of connecting to a remote session manager. This process is internal to NETEX. The user's connect is in progress.
<b>assigning</b>	A user is in the process of being identified as a network user. This state is internal to NETEX. The user's offer or connect is in progress.
<b>Rnref</b>	This field shows the destination (or remote) host's Nref for this network connection. If a connection does not currently exist, this column will be blank.
<b>Reads</b>	This field shows the number of user messages received during this session.
<b>Writes</b>	This field shows the number of user messages transmitted during this session.
<b>Nubblki</b>	Maximum input blocksize.
<b>Nubblko</b>	Maximum output blocksize.
<b>Readto</b>	Read timeout.

# DISPLAY PARMS

## Description

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

## Format

Display Parms

## Display Examples

The following figure shows a sample DISPLAY PARMS command output:

10:09:45	Host	ULTRA5H	Parameters
contime= 13	deadtime= 60	idletime= 6	readtime= 30
maxbi= 65400	maxbo= 65400	defbi= 32768	defbo= 32768
segsize=32768	wdogint= 2	msglvl=interest	mtudisc= 0
max ses= 110	max tran= 0	max net= 0	dreadque= 12
cur ses= 3	cur tran= 4	cur net= 4	status=NORMAL
lim ses= 255	rmtOper= ON	class= G	multihost= OFF
prefprot= 2	mxbks= 0	xdbg= 0	pollsel= ON
userexmitq= 1	rexmwbks= 2	bufolim= 2000	usercvgapq= 0
dtqhs= 15000	dtqls= 8000	dtqh= 20000000	dtql= 10000000
dbgmsg= 0	dbgdata= 0	dbgreg= 0	dbgret= 0
NCTVer= 128	mdbufin= 5	mdbufout= 5	
msgsyslog= 3	msgsyslogfac= local3		
max tnp= 0	cur tnp= 0		

Figure 28. Display Parms Command

The following describes the fields:

### contime

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

### deadtime

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

### idletime

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

### readtime

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

**maxbi**

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

**maxbo**

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

**defbi**

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

**defbo**

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

**segsiz**

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

**wdogint**

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

**msglvl**

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

**mtudisc**

0=don't , meaning set path mtu discovery off, so that IP does not set the 'DF' (don't fragment) bit  
1=want , meaning fragment the datagram if necessary, or set the 'DF' bit otherwise

2=do, meaning set path mtu discover on for STREAM sockets and force the 'DF' bit to be set.

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

**max tran**

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

**max net**

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



**dreadque**

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

**cur ses**

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

**cur tran**

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

**cur net**

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

**status**

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

**DRAINED**     A DRAIN command has been issued and no sessions are active.

**DRAINING**    A DRAIN command has been issued, but some sessions are still active.

**NORMAL**       All sessions are active.

**lim ses**

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

**rmtOper**

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

**class**

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

**A**            indicates that all commands are allowed.

**G**            indicates that only display commands are allowed.

**multihost**

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

**prefprot**

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

**mxkbs**

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

specified in Kbits per second. For example, a value of 50 means 50Kbs; a value of 50000 means 50Mbs (i.e. 50,000 Kbs).

#### **xdbg**

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

#### **pollsel**

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

#### **userexmitq**

Specifies whether or not to use the retransmit queue.

#### **rexmwblks**

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

#### **bufolim**

This is the maximum number of segments allowed waiting to be sent

#### **usercvgapq**

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

#### **dtqhs**

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

#### **dtqls**

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

#### **dtqh**

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

#### **dtql**

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

#### **msgsyslog**

This specifies where Netex msgs will be output. A value of 1 means output to the Netex log. A value of 2 means output to the system log. A value of 3 means output to both logs. A value of 0 means no syslog and minimal Netex log output.

## msgsyslogfac

This specifies how syslog msgs will be output. A default syslog facility code of “local3” will be used unless overridden.

The valid facility codes are system dependent. For Linux 2.6 they are:

- auth
- authpriv
- cron
- daemon
- ftp
- kern
- lpr
- mail
- mark
- news
- security
- syslog
- user
- uucp
- local0
- local1
- local2
- local3
- local4
- local5
- local6
- local7

It is recommended that only the local facility codes be used. Check with your system administrator for more information.

The syslog priority will be mapped to the Netex severity codes as follows:

Ntx Msg Severity		Syslog Priority	
blither	0	DEBUG	7
debug	1	DEBUG	7
monitor	2	INFO	6
moderate	3	NOTICE	5
interesting	4	WARNING	4
important	5	ERR	3
immediate	6	ALERT	1

## max tnp

This is the maximum number of tnp sessions allowed by the current license.

**cur tnp**

This is the current number of tnp sessions in progress.

**dbgmsg**

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

**dbgdata**

This is the number of data bytes being traced

**dbgreq**

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

**dbgret**

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

**NCTVer**

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

**mbufin**

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

**mbufout**

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

## DISPLAY SESSION

### Description

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

### Format

```
Display Session [sref]
```

### Parameters

#### sref

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

### Display Examples

The following shows a sample DISPLAY SESSION command output:

12:45:32 Host ULTRA5H Active Sessions											
Sref	User	Pid	State	Name	Host	Rnref	Msg In	Msg Out	Tid		
56	NTXOPER	-2	offered	NTXOPER					0000		
65	root	87008	data	ROOT	SOLSRVRH	133	7	8	0000		
66	root	00208	offered	NTXEFT					0000		

**Figure 29. Display Session, no SREF specified**

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

12:49:40	Host ULTRA5H	Sref 65									
User = root	State=data	Dest =SOLSRVRH	Rnref=	133							
Mxbi = 16384	Mxbo = 16384	Class=	2	Rate =	0						
Reads= 7	Writes= 8	Pid = 21587008	Name =								
Ruser=	Flag=	00	Tid =	0000							

**Figure 30. Display Session, SREF specified**

The following describes the fields in the display:

#### Sref

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

#### User

This field shows the username and process id requesting session services.

#### Pid

This field shows the process id of the user of this session. If pid = -2, this is the remote operator's session.

#### State

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

<b>closing</b>	A close has been issued by the user. No additional data may be sent, but additional data may be received.
<b>confirm</b>	CONNECT message received, waiting for the confirm call.
<b>data</b>	Connection completed and user may exchange data.
<b>connect</b>	Connect request issued by user, waiting for confirm.
<b>disconnect</b>	Disconnect detected, but not yet complete.
<b>offered</b>	Offer has been issued by user, waiting for connect.
<b>smgr conn</b>	A user is in the process of connecting to a remote session manager. This process is internal to NETEX. The user's connect is in progress.
<b>assigning</b>	A user is in the process of being identified as a network user. This state is internal to NETEX. The user's offer or connect is in progress.

### **Rnref**

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

### **Reads**

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

### **Writes**

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

# DISPLAY TRANSPORT

## Description

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

## Format

Display Transport [tref]

## Parameters

### tref

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

## Display Examples

The following shows a sample DISPLAY TRANSPORT command output:

14:01:50	Host BUDDY3	Active Transport Connections						
tref	username	segsz	state	rnref	blksin	duprcv	blkout	rexmit
2	root	32768	offered					
17	NTXOPER	32768	offered					
18	FARRERS	32768	data	19	63641	0	1	0
19	FARRERS	32768	data	18	1	0	63641	0
65535	SESSMGR	32768	offered					

Figure 31. Display Transport, no TREF specified

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

14:02:00	Host BUDDY3	Tref	19						
User =	FARRERS	Name =	NETEXEAT	State=	data	RmtNref=	18		
AckQ =	0	InQ =	5	OutQ =	0	DataQ=	0		
Maxtblok=	5	Mblk=	32000	Mrate=	0	Segsize=	32768		
Maxrblok=	5	Mblk=	32000	Reads=	1	Writs=	65850		
CurrKBS=	0	CurrMsRTDl=	0	PipeB=	160000	CurrOB=	160000		
MinRTDMs=	0	MaxRTDMs=	0	LclSnRt=	7488	LSRMax=	0		
Prot=	2	RcvKBS=	0	RmRcvRt=	0	RRRMax=	0		
RateP=	200	EquivP=	750	DtQB=	0	DtQHM=	0		
CurPmRt=	0	MaxNPDU=	32768	NrbMxRt=	0	DtQHMS=	1		
BufOut=	5	BufOLim=	2000	BufOLimC=	0	UseRcvGapQ=	0		
ReXmitQ=	0	UseReXmQ=	1	ReXmWBlks=	2	RcvGapQ=	0	0	
DtQHB=	20000000	DtQLB=	10000000	DtQHS=	15000	DtQLS=	8000		
Transmitter:		Olrn =	319	Plrn =	314	Opbn =	314		
		Rexmt=	0	Tack =	00000000	Tpbn =	314		
		Curtb =	09d6a540	Tto =	6	Ackcr=	2		
Receiver:		Ilrn =	2	Rplrn=	6	Rlrn =	0		
		Rdup =	0	Rack =	00000000	Rpbn =	1		
		Currb =	09d7aba8	Rto =	6	Ackcr=	2		
		PbnOO=	0						

### Figure 32. Display Transport, TREF specified

The following describes the fields in the displays:

#### **Tref**

This is the NETEX transport reference identifier.

#### **Username**

This is the name of the process requesting transport services.

#### **State**

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

<b>closing</b>	A close has been issued by the user. No additional data may be sent, but additional data may be received.
<b>confirm</b>	CONNECT message received, waiting for the confirm call.
<b>data</b>	Connection completed and user may exchange data.
<b>connect</b>	Connect request issued by user, waiting for confirm.
<b>disconnect</b>	Disconnect detected, but not yet complete.
<b>rdconnect</b>	Offer has been issued by user, waiting for connect.
<b>smgr conn</b>	A user is in the process of connecting to a remote session manager. This process is internal to NETEX. The user’s connect is in progress.
<b>assigning</b>	A user is in the process of being identified as a network user. This state is internal to NETEX. The user’s offer or connect is in progress.

#### **RmtNref**

This is the remote network reference number.

#### **AckQ**

This is the number of blocks queued waiting for acknowledgement.

#### **InQ**

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

#### **OutQ**

This is the number of segments queued to be sent.

#### **DataQ**

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

#### **Maxtblok**

This is the maximum number of transmitting buffers.

#### **Mblko**

This is the maximum output block size (bytes).

#### **MRate**

This is the maximum transmission rate in Kbits/sec.



**Segsize**

This is the transport layer segmentation size.

**Maxrblok**

This is the maximum number of receiving buffers.

**Mblki**

This is the maximum input block size (bytes).

**Reads**

This is the number of reads completed by transport.

**Writs**

This is the number of writes issued by transport.

**CurrKBS**

This is the current target transmission rate in KBytes/Sec

**CurrMsRTDI**

This is the current millisecond round trip delay.

**PipeB**

This is the current size of the pipe in bytes.

**CurrOB**

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

**MinRTDms**

This is the minimum round trip delay detected.

**MaxRTDms**

This is the maximum round trip delay detected.

**LclSnRt**

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

**LSRMax**

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

**Prot**

This is the protocol in use for this connection.

**RcvKBS**

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

**RmRcvRt**

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

**RRRMax**

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

**RateP**

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

#### **EquivP**

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

#### **DtQB**

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

#### **DtQHM**

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

#### **CurPmRt**

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

#### **MaxNPDU**

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

#### **NrbMxRt**

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

#### **DtQHMS**

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

#### **BufOut**

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

#### **BufOLim**

This is the maximum number of segments allowed waiting to be sent.

#### **BufOLimC**

This is the highest number of segments that were ever waiting to be sent.

#### **UseRcvGapQ**

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

#### **ReXmitQ**

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

#### **UseReXmQ**

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

#### **ReXmWBlks**

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

#### **RcvGapQ**

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

**DtQHB**

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

**DtQLB**

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

**DtQHS**

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

**DtQLS**

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

***Transmitter:***

**Oln**

This is the last LRN (Logical Record Number) assigned

**Plrn**

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

**Opbn**

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

**Rexmt**

This is the number of blocks retransmitted.

**Tack**

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

**Tpbn**

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

**Curtb**

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

**Tto**

This is the transmit timeout (idle time).

**Ackcr**

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

***Receiver:***

**Ilrn**

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

**Rplrn**

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

**Rlrn**

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

**Rdup**

This is the number of duplicate LRNs received.

**Rack**

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

**Rpbn**

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

**Currb**

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

**Rto**

This is the receive timeout (communications lost).

**Ackcr**

This is the incoming ACK credit.

**PbnOO**

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

## **DISPLAY USAGE**

### **Description**

See the “DISPLAY MEMORY” command on page 83.

## DISPLAY VERSION

### Description

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

### Format

Display Version

### Display Example

The following shows the output from a DISPLAY VERSION command for H800IP:

```
NtxOper> d version  
  
14:21:19          Host NETFIN10          Version  
H800IP LINUX NETEX Rel 7.0.0 2012/04
```

Figure 33. Display Version Command

## DRAIN ADAPTER

### Description

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

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

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

Link or gateway adapters may not be drained at this time.

### Format

DRAIN Adapter <uuss>

### Parameters

#### uuss

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

# DRAIN NETEX

## Description

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

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

## Format

```
DRAIN NETEX
```

## **DRAIN HOST**

### **Description**

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

### **Format**

DRAIN HOST rmthost

### **Parameters**

#### **rmthost**

Specifies the remote host to be drained.



## HALT SREF

### Description

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

### Format

Halt SRef n

### Parameters

**n**

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

# HELP and lhelp

## Description

- HELP Provides a list of NetEx/IP commands and command formats.
- ? Provides a list of NetEx/IP commands and command formats.
- lhelp Provides a list of NETEXOPER commands and command formats. lhelp must be lower case.

## Format

HELP  
?

## lhelp

## Parameters

None.

## Comments

This command can be entered as either 'help' or '?' for NetEx/IP commands. Enter 'lhelp' for NETEXOPER commands.

# **KILL NETEX**

## **Description**

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

## **Format**

```
KILL NETEX
```

## **LOAD KEY**

### **Description**

This command loads the license key located in the NESikeys

### **Format**

Load Key

## LOAD NCT

### Description

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

The operator must first modify the NCT data file containing the configuration statements and run the CM utility to create the PAM file (refer to “Step 4: Build an NCT” in the appropriate appendix for your system). The filename given on the MAKEPAM statement must be the same name given on the LOAD command. Also, the local hostname specified on the MAKEPAM statement must be the same as the local host in NETEX (the local parameter in the ntx\_default file). If the message “file not in PAMFILE format” is returned, check to see if your host name matches the NETEX local hostname.

### Format

Load Nct filename

### Parameters

#### filename

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

(/usr/share/nesi/netex/site/ntx\_pam)

## **SET BUFOLIM**

### **Description**

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

### **Format**

SET BUFOLim number

### **Parameters**

#### **number**

Specifies the number of buffers (segments).

## **SET CONTIME**

### **Description**

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

### **Format**

SET Contime seconds

### **Parameters**

#### **seconds**

Specifies the number of seconds that NETEX waits to generate a response to a transport connection message.

## SET DBGDATA

### Description

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

### Format

```
SET DBGDATA byte_count
```

### Parameters

#### byte\_count

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



## SET DBGMSG

### Description

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

### Format

SET DBGMSG value

### Parameters

#### value

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

## SET DBGREQ

### Description

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

### Format

SET DBGREQ value

### Parameters

#### value

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

## SET DBGRET

### Description

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

### Format

```
SET DBGRET value
```

### Parameters

#### value

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

## SET DEADTIME

### Description

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

### Format

SET DEadtime seconds

### Parameters

#### seconds

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

## **SET DEFBI**

### **Description**

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

### **Format**

SET DEFBI size

### **Parameters**

**size**

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

## SET DEFBO

### Description

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

### Format

SET DEFBO size

### Parameters

#### size

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

## SET IDLETIME

### Description

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

### Format

SET Idletime seconds

### Parameters

#### seconds

Specifies the number of seconds NETEX waits before sending an idle message to the remote host.

## SET IPROUTE

### Description

Create an IP routing table entry.

### Format

```
SET IProute uuss ipaddr
```

### Parameters

#### **uuss**

The NCT defined NETADDR and SMGDREF of a host adapter.

#### **ipaddr**

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

### Notes

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



## SET MAXBI

### Description

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

### Format

`SET MAXBI size`

### Parameters

#### **size**

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

# SET MAXBO

## Description

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

## Format

SET MAXBO size

## Parameters

### size

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

## SET MAXKBS

### Description

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

### Format

```
SET MAXKBS n
```

### Parameters

**n**

Specifies the maximum rate in Kbits per second. For example, a value of 50 means 50 Kbs; a value of 50000 means 50 Mbs (50,000 Kbs).

## SET MBUFIN

### Description

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

### Format

```
SET MBUFIN n
```

### Parameters

**n**

Specifies the number of buffers.

## SET MBUFOUT

### Description

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

### Format

```
SET MBUFOUT n
```

### Parameters

**n**

Specifies the number of buffers.

# SET MSGLVL

## Description

Controls the severity of messages printed on the operator's console. The operator messages are grouped from 0-15. All messages with the specified level of severity or greater are displayed.

## Format

SET MSGLvl level

## Parameters

### level

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

- immediate** Messages that require immediate action by the operator. Example: NETEX termination.
- important** Messages that are of great interest to the operator and may require operator action. Examples: notification of all set drain, start, and clear commands; remote operator messages.
- interesting** Messages regarding events that are of interest in closely monitored environments.
- blither** Messages that are intended for diagnostic or debugging purposes. These messages are generally only of interest when a system programmer is attempting to diagnose a NETEX problem.

## SET MSGSYSLOG

### Description

Set where Netex msgs will be output

### Format

```
SET MSGSyslog value
```

### Parameters

#### value

0, 1, 2, or 3. A value of 0 means minimal output to the Netex log and no output to syslog. A value of 1 means output to the Netex log only. A value of 2 means output to the system log. A value of 3 means output to both logs.

Values 0 and 1 are the only valid options for H140IP.

# SET MSGSYSLOGFAC

## Description

Set how Netex msgs will be output to the system syslog

This command is not applicable for H140IP.

## Format

```
SET MSGSYSLOGFac syslog_facility
```

## Parameter

### syslog\_facility

This specifies how syslog msgs will be output. A default syslog facility code of “local3” will be used unless overridden.

The valid facility codes are system dependent. For Linux 2.6 they are:

- auth
- authpriv
- cron
- daemon
- ftp
- kern
- lpr
- mail
- mark
- news
- security
- syslog
- user
- uucp
- local0
- local1
- local2
- local3
- local4
- local5
- local6
- local7

It is recommended that only the local facility codes be used. Check with your system administrator for more information.



The syslog priority will be mapped to the Netex severity codes as follows:

<b>Ntx Msg Severity</b>		<b>Syslog Priority</b>	
blither	0	DEBUG	7
debug	1	DEBUG	7
monitor	2	INFO	6
moderate	3	NOTICE	5
interesting	4	WARNING	4
important	5	ERR	3
immediate	6	ALERT	1

## SET MULTIHOST

### Description

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

### Format

```
SET MULTIhost value
```

### Parameters

#### value

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

## SET NTXOPER

### Description

Specifies whether the remote operator service is enabled or disabled.

### Format

```
SET NTXOPer value
```

### Parameters

#### value

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

## SET POLLSEL

### Description

Specifies whether to poll with no timeout or not.

### Format

```
SET POLLSel {on | off}
```

### Parameters

**on | off**

# SET PREFPROT

## Description

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

## Format

```
SET PREFPROT n
```

## Parameters

**n**

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

## SET RCVDATAQHB

### Description

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

### Format

SET RCVDATAQHB number

### Parameters

#### **number**

Specifies the number of bytes.

## SET RCVDATAQLB

### Description

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

### Format

SET RCVDATAQLB number

### Parameters

#### number

Specifies the number of bytes.

## SET RCVDATAQHS

### Description

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

### Format

SET RCVDATAQHS number

### Parameters

#### number

Specifies the number of segments.



## SET RCVDATAQLS

### Description

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

### Format

SET RCVDATAQLS number

### Parameters

#### number

Specifies the number of segments.

## SET REXMWBLKS

### Description

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

### Format

SET REXMWBlks number

### Parameters

#### number

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

## SET READTIME

### Description

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

### Format

```
SET READtime seconds
```

### Parameter

#### seconds

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

## SET ROPCLASS

### Description

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

### Format

```
SET ROPClass class
```

### Parameters

#### class

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

A indicates that all commands are allowed.

G indicates that only display commands are allowed.

# SET SESMAX

## Description

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

## Format

SET SESMax number

## Parameters

### number

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

## **SET USERCVGAPQ**

### **Description**

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

### **Format**

```
SET USERCVgapq { 0 | 1 }
```

### **Parameters**

**0 | 1**

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

## SET USEREXMITQ

### Description

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

### Format

```
SET USEREXmitq { 0 | 1 }
```

### Parameters

**0 | 1**

0 means do not use the retransmit queue, 1 means use it.

## SET WDOGINT

### Description

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

### Format

```
SET WDogint seconds
```

### Parameters

#### seconds

Specifies the number of seconds that NETEX uses as a base unit for checking time out values.

## START NETEX

### Description

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

### Format

```
START NETEX
```



## START ADAPTER

### Description

Start a DRAINED adapter.

### Format

STart Adapter <uuss>

### Parameters

#### **uuss**

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

## **START HOST**

### **Description**

Starts a remote host which had been drained.

### **Format**

```
START HOST rmthost
```

### **Parameters**

#### **rmthost**

Specifies the name of the remote host to be started.

# Appendix A: NRB Error Codes

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

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

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

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

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

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

**Table 2. Origin of NRB Error Codes**

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

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

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

**Table 3. Origin of NRB Error Codes, Part 2**

Note the following when using these codes:

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

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

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

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

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

## General Errors

The following errors are general NETEX errors.

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

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

**Table 4. General NRB Error Codes**

## Host Specific Errors

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

**Table 5. Host Specific NRB Error Codes**

## License Specific Errors

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

**Table 6. License Specific NRB Error Codes**

## Driver Service Errors

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

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

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

**Table 7. Driver Service NRB Error Codes**



## Transport Service Errors

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

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

**Table 8. Transport Service NRB Error Codes**

## Session Service Errors

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

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

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

**Table 9. Session Service NRB Error Codes**

## Network Service Errors

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

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

**Table 10. Network Service NRB Error Codes**





# Appendix B: Messages

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

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

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

## Messages:

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

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

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

### User Response:

None.

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

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

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

### User Response:

None.

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

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

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

**User Response:**

None.

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

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

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

**User Response:**

None.

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

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

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

**User Response:**

None.

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

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

- n identifies the internal NetEx/IP session reference number of the local message.
- rr identifies the NetEx/IP request type.
- ssss identifies the reason for the session failure. Refer to “Appendix A: NRB Error Codes” on page 147 for a description of the possible status codes.
- i identifies the status of the connection
  - 6 session is disconnected

**User Response:**

None.

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

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

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

**User Response:**

None.

<b>NETEX: network protocol error n</b>
--

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

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

**User Response:**

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

**NETEX: transport protocol error n**

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

- 1 -- base field too small
- 2 -- protocol level is not greater than or equal to 2
- 3 -- invalid message type
- 4 -- connect attempt to unlicensed PROT\_4
- 5 -- tdb->tdblrn > tub->tubrplrn
- 6 -- ‘disc’ sub-field in ‘x’ msg type, ‘type’
- 7 -- invalid sub-field type ‘x’, ‘type’
- 8 -- sub-field ‘x’ too small = ‘y’, ‘type’, ‘size’
- 9 -- sub-field ‘x’ length ‘y’ overlaps total transport length ‘z’, ‘type’, ‘size’, ‘length’
- 10 -- New path has smaller segsize

**User Response:**

None. Netex reports the invalid message and continues.

**Message received with no read active**

**Description:** A Netex message was received on the network, but there was no Netex ‘read’ request active at the time. Message is dropped.

**User Response:**

None. Netex reports the condition and continues.

Repeated occurrences of this message could indicate the Netex ‘DREADQUEUE’ parameter might need to be increased.

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

**Explanation:** The Netex license initialization has failed. “cccc” specifies the reason code for the failure:

- 9001 : expiring
- 9002 : expired
- 9003: expired – product non-functional

9004 : license open error

9005 : invalid key

9007 : no fingerprint

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

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

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

**Explanation:** Specifies the reason code for the failure:

9001 : expiring

9002 : expired

9003: expired – product non-functional

9004: key file not found

9005 : invalid key

9007 : no fingerprint

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

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





# Appendix C: Running Multiple NetExes

## Atypical Operations

The standard implementation of a NetEx configuration on a UNIX or Windows server is a single instance of NetEx running and the applications using the standard API library to use this instance of NetEx. There are reasonable situations where it may be advantageous to have multiple instances of NetEx running on a single UNIX or Windows server. One may be for redundancy in a clustered environment. Another would be certain NetEx connections from a single server that might require different transport parameters to run efficiently.

This section outlines what needs to be done to run multiple NetExes on the UNIX or Windows server. It is recommended that customers review their implementation plan with NESi Support at [support@netex.com](mailto:support@netex.com) prior to going to production.

This section explains which NetEx parameters and shell environment overrides need to be configured to support multiple instances of NetEx on a single server.

## Required Shell Environment Variables

NetEx will interpret the following shell environment variable to determine which file to use when starting NetEx:

- `NTXDEFAULT` – The fully qualified filename of the `ntx_default` file to be used when starting this NetEx instance. The default file contains startup parameters for this NetEx.

NetEx applications (via the NetEx API library) interpret the following shell environment variable to select the particular NetEx to use:

- `NETEX_DEVICE` – Set to the named socket of the NetEx an application wants to use for data transfer. This is the `ntx_default` mailbox setting described below.

## Required `ntx_default` File Parameter Settings

- `Local` – Defines a unique NetEx hostname for this NetEx instance
- `Mbxname` – Defines a uniquely named mailbox this NetEx will use to communicate with applications
- `Pamfile` – The pam file created for this NetEx instance hostname.
- `ip1intrf` – Set to 0 (zero) so this NetEx binds to specific IP address(es).
- `Device` – Defines the unique GNA address(es) to use for this NetEx instance

## Suggested `ntx_default` File

`Ntxlogname` – Defines the NetEx log name to use for this NetEx instance. It is suggested that each NetEx instance have its own log.

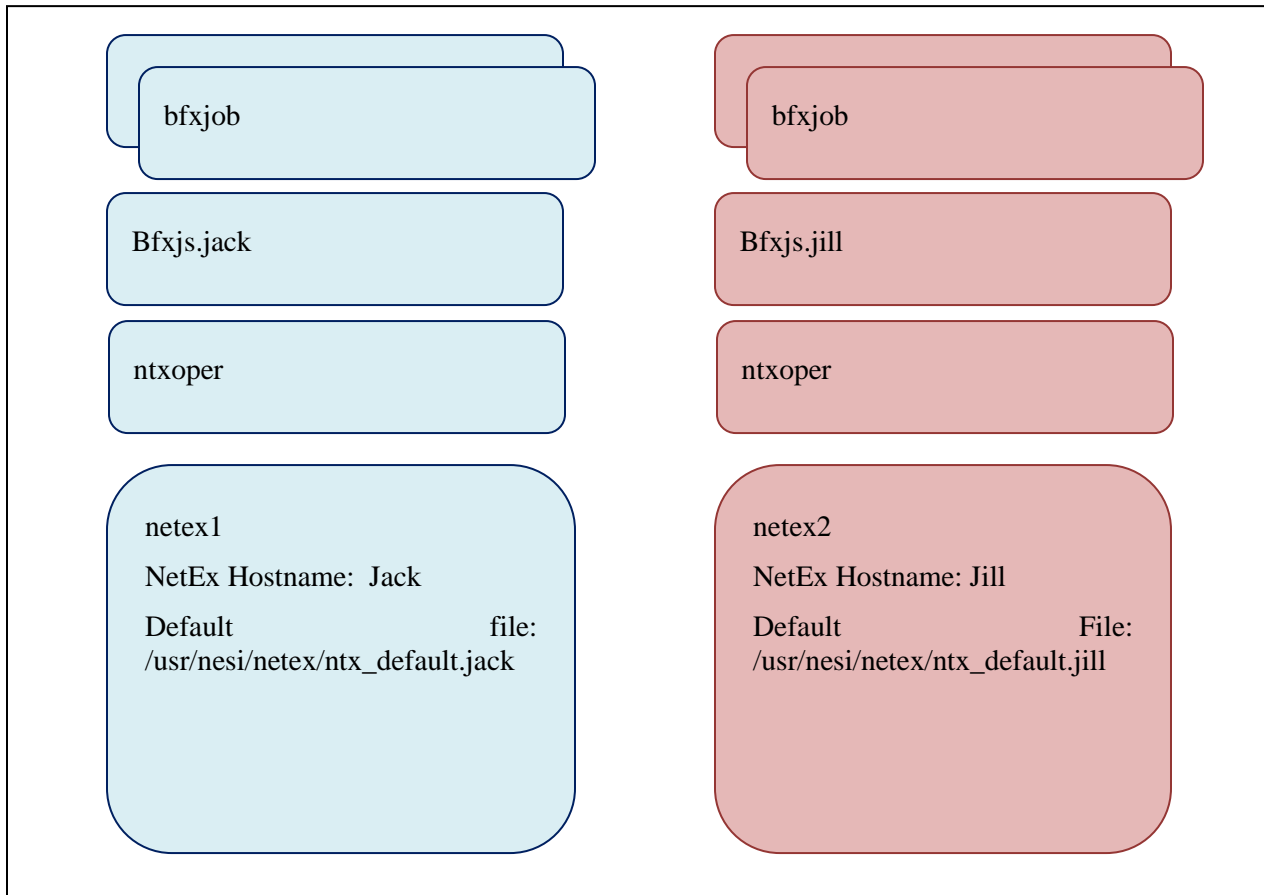
## Order of Events For Multiple NetEx Instances On a Server:

1. Configure a `ntx_default` file for each NetEx instance
2. Define `NTXDEFAULT` shell environment variable.
3. Start `netex`
4. Define shell environment variable `NETEX_DEVICE` for the desired NetEx
5. Start application(s) (i.e. `ntxoper`, `bfxjs`, etc.)
6. Repeat steps 1-3 for additional NetEx instances.

7. Repeat steps 4 and 5 for additional jobs.

## Example of Two Netexes on a Server

The following is a diagram symbolizing a server with two Netexes, each with their own bfx jobs, bfxjs and ntxoper applications.



Note the applications using NetEx do not have to be uniquely named, just that the appropriate `NETEX_DEVICE` shell environment variable needs to be set for the desired NetEx. The exception to this is `bfxjs` as the `bfxstart` script checks if `bfxjs` is already running. For ease of tracking started jobs or troubleshooting the job instances may be uniquely named to identify their preferred NetEx instance.

The following is illustrates the requirements for the above example for the `ntx_default` file entries and environment variables for `netex1` and `netex2` when defining them to run on a single host, or in a cluster where running on a single host may occur.

## Netex1

### 1. `/usr/nesi/netex/site/Ntx_default.jack` file entries:

```
local      jack
mbxname    /var/run/netexserver.jack
pamfile    /usr/nesi/netex/site/ntx_pam.jack
ntxlogname /var/opt/netex/ntx.jack.log
ip1intrf   0
device1    UDP-6950-00001704
```

### 2. When starting the `netex1` instance set the following shell environment parameter:

```
NTXDEFAULT=/usr/nesi/netex/site/ntx_default.jack
```

(i.e. “`export NTXDEFAULT=/usr/nesi/netex/site/ntx_default.jack`” after login and before starting NetEx)

### 3. File `/usr/nesi/netex/site/ntx_default.jack` will be used as the default file during NetEx startup

```
/usr/nesi/netex/ntx &
```

### 4. To start a NetEx application to use the `netex1` instance, set the following shell environment variable:

```
NETEX_DEVICE=/var/run/netexserver.jack
```

(i.e. “`export NETEX_DEVICE=/var/run/netexserver.jack`”)

### 5. You can then start the application and `jack` will be used for the data transfer. For example:

```
/usr/nesi/bfx/bin/bfxstart .jack
```

## Netex2

### 1. `/usr/nesi/netex/site/ntx_default.jill` file entries:

```
local      jill
mbxname    /var/run/netexserver.jill
pamfile    /usr/nesi/netex/site/ntx_pam.jill
ntxlogname /var/opt/netex/ntx.jill.log
ip1intrf   0
device1    UDP-6950-00001804
```

### 2. When starting the `netex2` instance set the following shell environment parameter:

```
NTXDEFAULT=/usr/nesi/netex/site/ntx_default.jill
```

### 3. File `/usr/nesi/netex/site/ntx_default.jill` will be used as the default file during NetEx startup

### 4. To start a NetEx application to use the `netex2` instance, set the following shell environment parameter:

```
NETEX_DEVICE=/var/run/netexserver.jill
```

5. You can then start the application and jill will be used for the data transfer.

## Other Considerations

When monitoring NetEx using ntxoper commands, the NETEX\_DEVICE environment variable must be set appropriately to view the specific NetEx instance.

Automated jobs submitted to NetEx must also set the NETEX\_DEVICE parameter to use the proper NetEx.

In the event a remote host job starts a NetEx application on the server where multiple NetEx instances are running, the remote host job must set the appropriate NETEX\_DEVICE parameter on the multi-NetEx server.

It may be more convenient to use a shell script to wrap these functions and name it for the appropriate NetEx. i.e. ntxoper.jill sets the variable prior to the actual ntxoper call.

If the standard NetEx and NetEx application startup procedures will be used, they will have to be modified to start, stop and monitor the specific Netex or the NetEx application. Bfxstart and bfxstop scripts must be modified to look for the appropriate bfxjs started for each instance of NetEx.

If the NetEx log names have changed for this deployment, the scripts for managing the NetEx logs will have to be adjusted as well.

Using unique process names for each NetEx (i.e. symbolic links for each name to ntx) will assist in limiting confusion when monitoring the system process statuses.

# Appendix D: H800IP Linux Installation

## Prerequisites

The following are hardware and software prerequisites for installing the H800IP product.

- An Intel compatible system running a supported Linux OS. Review the website for supported OS distributions.
- At least one other processor on the network running NetEx/IP software. This processor should be connected with another NetEx/IP (not required for intra-host test/evaluation).
- Customers must obtain a software KEY from NESi prior to running the H800IP software. Customers must contact NESi customer support with the customer site name, hostname, and the NetEx/IP product designator (e.g., H800IP). NESi customer support will supply the necessary key once this information has been received. The customer needs to place this key into the NESikeys file as discussed later in this section.

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

## Hardware Installation

Install and verify proper operation of the appropriate operating system.

## Accessing the H800IP software distribution

The H800IP NetEx/IP software is available as an RPM which may be downloaded from NESi. Contact NESi Customer Support to request the download link.

## Getting the NESi Public Key

The RPM software distribution package is signed to ensure integrity and authenticity. It is recommended to install the NESi public key and verify the signature of any software packages before installation.

You can download the key from <http://www.netex.com/files/1813/7123/4715/RPM-GPG-KEY-netex.txt>.

## Importing the NESi Public Key

Install the public key as super user with the command:

```
# rpm --import RPM-GPG-KEY-netex.txt
```

## Verifying Signatures

You can verify the RPM signature to ensure that a package has not been modified since it has been signed. Verification will also check that a package is signed by the vendors or packagers key.

To verify the signature, use the `-K` or `--checksig` option to the `rpm` command:

```
# rpm -K H800IP-7.0.1-1.i386.rpm
```

## Software Installation

If this is an initial installation, install the software as super user with the command:

```
# rpm -i H800IP-7.0.1-1.i386.rpm
```

If the NESi public key has not been installed use the command:

```
# rpm -i --nosignature H800IP-7.0.1-1.i386.rpm
```

## Upgrading H800IP

If you are upgrading an existing installation of H800IP, it is strongly recommended that any running NetEx/IP process be stopped. *If you are upgrading from a release which is older than the most recent previous release you should save the configuration files (NCT, PAM and ntx\_default), remove or uninstall, and perform the instructions for an initial install.* Using the “rpm -U” command preserves any customized files in this package and the replacement files are installed with extensions of “.rpmnew”. Any files that are not in the package but in package directories will also be preserved. Upgrade the software as super user with the command:

```
# rpm -U H800IP-7.0.1-1.i386.rpm
```

If the NESi public key has not been installed use the command:

```
# rpm -U --nosignature H800IP-7.0.1-1.i386.rpm
```

## Removing H800IP

If H801 is installed it should be removed prior to the removal of H800IP.

During RPM removal, any customized files and log files (e.g., **ntx\_default**, **ntx\_pam**, **prodconf**, **NESikeys**, etc.) will not be deleted. Remove the software as super user with the command:

```
# rpm -e H800IP
```

## Removing the NESi Public Key

To remove the NESi public key, as super user issue the command:

```
# rpm -e gpg-pubkey-3d6b35d3-51bb5907
```

## Starting, Stopping & Verifying Install of Netex

The service command should be used to stop & start BFX:

```
# service netex stop
# service netex start
# service netex restart
```

The chkconfig command should be used to verify installation:

```
# chkconfig --list netex
```

# Post Installation Considerations

## Configuring H800IP

Once the software package installation has been successfully completed, NetEx/IP must be configured prior to execution. The following instructions address editing the files associated with product activation (`/usr/share/nesi/netex/site/prodconf` and `/usr/share/nesi/NESikeys`), the configuration file for NetEx/IP (`/usr/share/nesi/netex/site/ntx_default`), creating a Network Configuration Table (NCT), creating the Physical Address Map file (PAM file), and starting the NetEx/IP process.

Configuring the H800IP NetEx/IP software consists of the following steps:

- Step 1 Edit the 'NESikeys' file
- Step 2 Edit the NTX\_DEFAULT file
- Step 3 Build an NCT
- Step 4 Create NetEx/IP addressing information
- Step 5 Create Code Conversion Table (optional)
- Step 6 Start NetEx
- Step 7 Verify that 'ntx' Starts Automatically On Reboot
- Step 8 Setup syslog file definition & logrotate rules

### Step 1. Edit the 'NESikeys' file

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

- The default file name is *NESikeys*.
- The LICPATH keyword/value pair in the product configuration file (see `/usr/share/nesi/netex/site/prodconf`) specifies the full path name to this file. The default is: */usr/share/nesi/NESikeys*.
- A leading '#', '\*', or ';' character, in a file record denotes a comment line.
- The systems programmer installing this software must edit this file to add a new encrypted Software Key each time such a key is obtained from NESi for H800IP and/or other license-enabled NESi products. This should be done prior to installing the product, and must be done prior to any attempt to run the product successfully.
- To obtain a key from NESi, contact NESi support, supplying the hostname of the machine the software is to be installed on. The hostname may be obtained by issuing the Linux command "*hostname*" with no parameters.
- The file may contain multiple keys per product due to new product releases or a change to the platform's fingerprint (on UNIX this corresponds to the hostname for the target host). If there are multiple keys the NESi product will use the first key found that matches the product name and system fingerprint starting at the top of the file. This makes it important to add new keys for an existing system to the top of the file. For example, if a new key is installed that provides a license date extension, or adds a new feature, adding this new key to the file before the old key ensures the new key will be used rather than the old key. To make the file easier to maintain over time, it is recommended that you precede each Key entry with a

comment line that documents the product designator (e.g., H800IP) and release level of the product that the key is associated with. It will then be easier to delete older Keys that are no longer valid for the product.

- The following shows an example of what a *NESIkeys* file might look like after adding several Keys to the file:

```
# Network Executive Software, Inc. Software License Key file
# Key for H800IP R7.0.1:
CGGZ-4AAA-AAAE-IAO5-O5OJ-SBHX-AUZ5-PL4D
```

## Step 2. Edit the NTX\_DEFAULT file

See Appendix I: NetEx Default File Configuration on page 205.

## Step 3. Build an NCT

You must customize the Network Configuration Table (NCT) for your system and network. As distributed, H800IP does not provide an NCT for your site. You must create the file locally on the host and transfer the file to NetEx/IP in order to access the network configuration.

To generate the NCT, perform the following tasks:

1. Move to the site directory by entering the following commands:

```
# cd /usr/share/nesi/netex/site
```

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

A sample file (nct.txt.samp) is provided with H800IP in the site directory.

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

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

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

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

```
# ./cm
Config>
Config> nct sample.conf
Config> select *
Config> makepam local_hostname /usr/share/nesi/netex/site/ntx_pam
Config> exit
```



## Step 4. Create NetEx/IP addressing information

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

### 1. Update DNS nameserver information.

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

```
NTX0000uuss
```

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

### 2. Update local host table.

This method requires that you update the local host table (usually located at '/etc/hosts') with the IP address and IP hostname (same format as in #1).

### 3. Use the *SET IPROUTE* command.

The third option is to update the NetEx/IP network information via the new NTXOPER command **SET IPROUTE**. Please refer to the 'Chapter 8: Operator Interface' section of the "*NetEx/IP for UNIX or Windows Systems*" manual for a description of the command. These commands may be placed in the *ntx\_default* file after the 'ENDINI' statement so that they are executed on every startup.

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

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

These addresses comprise the HYPERChannel *toGNA* addresses, as defined in the NetEx/IP NCT by the NETADDR (*uu*) and SMGDREF (*ss*) parameters.

## Step 5. Create Code Conversion Table (optional)

The code conversion table can now be modified without the need for the product source code or a recompilation of the software.

Customers can execute the command "*cctbld filename*" (where filename is a file location on the host) to create a text file version that they can edit. NetEx will look for the code conversion table at /usr/share/nesi/netex/site/cctable. However, if the customer creates the file in an alternate location, they must add the variable "cctable" to the "NTX\_DEFAULT" file along with the fully qualified pathname of the file.

NetEx/IP checks for the existence of a code conversion file during startup. If an alternate code conversion table does not exist, NetEx/IP will use the built-in default table.

## Step 6. Start NetEx

On an initial install NetEx will not start automatically nor will it start following an update. To Start NetEx use the following command:

```
service netex start
```

## Step 7. Verify that 'ntx' Starts Automatically On Reboot

NetEx has been configured to automatically start for run levels 3, 4, and 5 after a system reboot. It will not work until the 'ntx\_default' and 'ntx\_pam' files have been properly set up and placed in the correct locations, and the key is correct and in the correct location.

**Note:** If NetEx is started/stopped manually, the following script should be used as it properly modifies some system parameters required by NetEx and detects common problems:

```
service netex start | stop | restart | status
```

## Step 8. Setup syslog file definition & logrotate rules

Netex will send most messages to syslog using the facility 'local3' or what is defined in the ntx\_default file. The messages will have a syslog priority that is mapped to a Netex message severity. See "msgsyslogfac" for more details.

An example of the syslog definitions for creating netex logs from syslog as defined in /usr/share/nesi/netex/site/ntx.syslog.cnf is provided:

```
# rsyslog config for netex
local3.* /var/log/ntxloc3.log
```

The Netex logs may be rotated using your systems log rotation facility. An example of the file /etc/logrotate.d/ntxlogs.logrotate is also provided:

```
# netex logrotate config:
# rotate product log files
/usr/share/nesi/netex/site/ntxlog /usr/share/nesi/netex/site/tnp.log
/usr/share/nesi/netex/site/tnpntx.log {
    daily
    compress
    nocreate
    copytruncate
    nodateext
    maxage 365
    rotate 32
    missingok
    #notifempty
    ifempty
    create 644 root root
}

# rotate Netex syslog files
/var/log/ntxloc3.log /var/log/ntxloc4.log
    daily
    compress
    nocreate
    copytruncate
    nodateext
    maxage 365
    rotate 32
    missingok
    #notifempty
    ifempty
    create 644 root root
}
```

# Appendix E: H620IP AIX Installation

## Prerequisites

The following are hardware and software prerequisites for installing the H620IP product.

- An IBM Power System or System p server running, AIX® 5.3 to 7.1 distributions.
- At least one other processor on the network running NetEx/IP software. This processor should be connected with another NetEx/IP (not required for intra-host test/evaluation).
- Customers must obtain a software KEY from NESi prior to running the H620IP software. Customers must contact NESi customer support with the customer site name, hostname, and the NetEx/IP product designator (e.g., H620IP). NESi customer support will supply the necessary key once this information has been received. The customer needs to place this key into the NESikeys file as discussed later in this section.

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

## Hardware Installation

Install and verify proper operation of the AIX® system.

## Accessing the H620IP software distribution

The H620IP NetEx/IP software is available as an RPM which may be downloaded from NESi. Contact NESi Customer Support to request the download link.

## Software Installation

All installation steps must be completed by a user logged on as root.

Version 7.0.1 is the first version of H620IP NetEx/IP that installs from an RPM package.

## Upgrading H620IP if RPM was not used for the installation

If you are upgrading from a non-RPM distributed H620IP installation, you should terminate NetEx with the "ntxoper kill netex" command and save any files you used in the configuration and running of NetEx. These would typically include files located in /usr/nesi, such as the NESikeys file, the NCT file, and the ntx\_default files in /usr/nesi/netex/site/. In addition, the keys associated with any non-RPM releases are not compatible with the versions that install from an RPM package.

The script **/usr/nesi/DEINSTALL.netex** will remove the software from the system. If the /etc/inittab file was updated to start NetEx when the system was booted, you should remove these old NetEx entries. These may assist:

```
lsitab -a          (List all entries in the /etc/inittab)
```

Look for an entry that contains rc.netex. Look at the first parameter separated by a colon.

```
rmitab characters in the first parameter of the netex itab entry
```

Remove the current version of H621 (BFX for AIX).

```
# cd /usr/nesi
# rm -rf bfx
```

H621 (BFX for AIX) must be reinstalled with the current release.

Install the software as super user with the command:

```
# rpm -ivh H620IP-7.0.1-1.ppc.rpm
```

## Upgrading H620IP if RPM was used to install

If you are upgrading an RPM installation of H620IP, it is strongly recommended that any running NetEx/IP process be stopped. Using the “rpm -U” command preserves any customized files in this package and the replacement files are installed with extensions of “.rpmnew”. Any files that are not in the package but in package directories will also be preserved. Upgrade the software as super user with the command:

```
# rpm -U H620IP-7.0.1.-1.ppc.rpm
```

## Removing H620IP RPM

If H621 BFX was installed, this RPM for BFX must be removed first.

If you wish to remove the rpm, you may use the following command using your superuser id.

```
# rpm -e H620IP
```

During RPM removal, any customized files and log files (e.g., **ntx\_default**, **ntx\_pam**, **prodconf**, **NESikeys**, etc.) will not be deleted.

## Starting, Stopping & Verifying Install of NetEx

The startsrc command or SMIT should be used to start Netex:

```
# startsrc -s netex
```

The stopsrc command or SMIT should be used to stop Netex:

```
# stopsrc -s netex
```

The lssrc and lsitab commands or SMIT can be used to verify installation:

```
# lssrc -S -s netex
# lsitab netex
```

# Post Installation Considerations

## Configuring H620IP

Once the software package installation has been successfully completed, NetEx/IP must be configured prior to execution. The following instructions address editing the files associated with product activation (`/usr/share/nesi/NESikeys`), the configuration file for NetEx/IP (`/usr/share/nesi/netex/ntx_default`), creating a Network Configuration Table (NCT), creating the Physical Address Map file (PAM file), and starting the NetEx/IP process.

Configuring the H620IP NetEx/IP software consists of the following steps:

**Error! Reference source not found.**

Step 2 Edit the NTX\_DEFAULT file

Step 3 Build an NCT

Step 4 Create NetEx/IP addressing information

Step 5 Create Code Conversion Table (optional)

Step 6 Starting / Stopping NetExStarting / Stopping NetEx

Step 7 Verify that 'ntx' Starts Automatically On Reboot

Step 8 Setup syslog file definition & logrotate rules (optional)

### Step 1. Edit the 'NESikeys' file

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

- The default file name is `/usr/share/nesi/NESikeys`.
- A leading '#', '\*', or ';' character, in a file record denotes a comment line.
- The actual key must reside on a single line by itself and start in column 1.
- The systems programmer installing this software must edit this file to add a new encrypted Software Key each time such a key is obtained from NESi for H620IP and/or other license-enabled NESi products. This should be done prior to installing the product, and must be done prior to any attempt to run the product successfully.
- To obtain a key from NESi, contact NESi support, supplying the hostname of the machine the software is to be installed on. The hostname may be obtained by issuing the Linux/AIX command "`hostname`" with no parameters.
- The file may contain multiple keys per product due to new product releases or a change to the platform's fingerprint (on UNIX this corresponds to the hostname for the target host). If there are multiple keys the NESi product will use the first key found that matches the product name and system fingerprint starting at the top of the file. This makes it important to add new keys for an existing system to the top of the file. For example, if a new key is installed that provides a license date extension, or adds a new feature, adding this new key to the file before the old key ensures the new key will be used rather than the old key. To make the file easier to maintain over time, it is recommended that you precede each Key entry with a comment line that documents the product designator (e.g., H620IP) and release level of the product that

the key is associated with. It will then be easier to delete older Keys that are no longer valid for the product.

- The following shows an example of what a *NESikeys* file might look like after adding a key to the file. The first two lines are comments with the key on the third line. Additional comments and keys could be added.

```
# Network Executive Software, Inc. Software License Key file
# Key for H620IP R7.0:
CGGZ-4AAA-AAAE-IAO5-O5OJ-SBHX-AUZ5-PL4D
```

## Step 2. Edit the NTX\_DEFAULT file

See Appendix I: NetEx Default File Configuration on page 205.

## Step 3. Build an NCT

You must customize the Network Configuration Table (NCT) for your system and network. As distributed, H620IP does not provide an NCT for your site. You must create the file locally on the host and transfer the file to NetEx/IP in order to access the network configuration.

To generate the NCT, perform the following tasks:

1. Move to the `/usr/share/nesi/netex/site` directory by entering the following commands:

```
# cd /usr/share/nesi/netex/site
```

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

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

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

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

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

```
# ./cm
Config>
Config> nct sample.conf
Config> select *
Config> makepam netex_host_name ntx_pam
Config> exit
```

## Step 4. Create NetEx/IP addressing information

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

### 1. Update DNS nameserver information.

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

```
NTX0000uuss
```

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

### 2. Update local host table.

This method requires that you update the local host table (usually located at '/etc/hosts') with the IP address and IP hostname (same format as in #1).

### 3. Use the *SET IPROUTE* command.

The third option is to update the NetEx/IP network information via the new NTXOPER command **SET IPROUTE**. Please refer to the 'Chapter 8: Operator Interface' section of the "*NetEx/IP for UNIX or Windows Systems*" manual for a description of the command. These commands may be placed in the *ntx\_default* file after the 'ENDINI' statement so that they are executed on every startup.

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

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

These addresses comprise the HYPERChannel *toGNA* addresses, as defined in the NetEx/IP NCT by the NETADDR (*uu*) and SMGDREF (*ss*) parameters.

## Step 5. Create Code Conversion Table (optional)

The code conversion table can now be modified without the need for the product source code or a recompilation of the software.

Customers can execute the command "*cctbld filename*" (where filename is a file location on the host) to create a text file version that they can edit. NetEx will look for the code conversion table at /usr/share/nesi/netex/site/cctable. However, if the customer creates the file in an alternate location, they must add the variable "cctable" to the "NTX\_DEFAULT" file along with the fully qualified pathname of the file.

NetEx/IP checks for the existence of a code conversion file during startup. If an alternate code conversion table does not exist, NetEx/IP will use the built-in default table.

## Step 6. Starting / Stopping NetEx

On an initial install NetEx will not start automatically nor will it start following an update. To Start NetEx you may use the SMIT tool on AIX. Netex is defined as an AIX subsystem, and may be started or stopped through the SMIT interface. It may also be manually started or stopped with:

```
startsrc -s netex
stopsrc -s netex
```

## Step 7. Verify that 'ntx' Starts Automatically On Reboot

NetEx has been configured to automatically start for run levels 3, 4, and 5 after a system reboot. It will not work until the 'ntx\_default' and 'ntx\_pam' files have been properly set up and placed in the correct locations, and the key is correct and in the correct location. To permanently disable this, the super user may issue the command "rmitab netex".

## Step 8. Setup syslog file definition & logrotate rules (optional)

NetEx will send most messages to syslog using the facility 'local3' or what is defined in the ntx\_default file. The messages will have a syslog priority that is mapped to a NetEx message severity. See "msgsyslogfac" for more details.

For AIX, you should update the /etc/syslog.conf file for NetEx with:

```
local3.info /var/log/ntxloc3.log rotate size 1M time 1d files 7 compress
```

This will send all informational messages (or higher) to the ntxloc3 file. A new log will be created every day, or when the current size exceeds one megabyte. Seven versions of this file will be retained.

To manage the ntxlog located in /usr/share/nesi/netex/site, the user may wish to implement a cron job to issue the NTXOPER SWLOG command on a daily basis. The number of logs retained is controlled by the ntx\_default parameter "numlogs".



# Appendix F: H820IP Linux for System z Installation

## Prerequisites

The following are hardware and software prerequisites for installing the H820IP product.

- An IBM System z running Red Hat® Enterprise Linux® 6.4 for IBM System z.
- At least one other processor on the network running NetEx/IP software. This processor should be connected with another NetEx/IP (not required for intra-host test/evaluation).
- Customers must obtain a software KEY from NESi prior to running the H820IP software. Customers must contact NESi customer support with the customer site name, hostname, and the NetEx/IP product designator (e.g., H820IP). NESi customer support will supply the necessary key once this information has been received. The customer needs to place this key into the NESikeys file as discussed later in this section.

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

## Hardware Installation

Install and verify proper operation of the appropriate operating system.

## Accessing the H820IP software distribution

The H820IP NetEx/IP software is available as an RPM which may be downloaded from NESi. Contact NESi Customer Support to request the download link.

## Getting the NESi Public Key

The RPM software distribution package is signed to ensure integrity and authenticity. It is recommended to install the NESi public key and verify the signature of any software packages before installation.

You can download the key from <http://www.netex.com/files/1813/7123/4715/RPM-GPG-KEY-netex.txt>.

## Importing the NESi Public Key

Install the public key as super user with the command:

```
# rpm --import RPM-GPG-KEY-netex.txt
```

## Verifying Signatures

You can verify the RPM signature to ensure that a package has not been modified since it has been signed. Verification will also check that a package is signed by the vendors or packagers key.

To verify the signature, use the `-K` or `--checksig` option to the `rpm` command:

```
# rpm -K H820IP-7.0.1-1.e16.s390.rpm
```

## Software Installation

If this is an initial installation, install the software as super user with the command:

```
# yum install H820IP-7.0.1-1.el6.s390.rpm
```

or

```
# rpm -i H820IP-7.0.1-1.el6.s390.rpm
```

If the NESi public key has not been installed use the command:

```
# yum --nogpgcheck install H820IP-7.0.1-1.el6.s390.rpm
```

or

```
# rpm -i --nosignature H820IP-7.0.1-1.el6.s390.rpm
```

## Upgrading H820IP

If you are upgrading an existing installation of H820IP, it is strongly recommended that any running NetEx/IP process be stopped. Using the “rpm -U” command preserves any customized files in this package and the replacement files are installed with extensions of “.rpmnew”. Any files that are not in the package but in package directories will also be preserved. Upgrade the software as super user with the command:

```
# yum upgrade H820IP-7.0.1-1.el6.s390.rpm
```

or

```
# rpm -U H810IP-7.0.1-1.el6.s390.rpm
```

If the NESi public key has not been installed use the command:

```
# yum --nogpgcheck upgrade H820IP-7.0.1-1.el6.s390.rpm
```

or

```
# rpm -U --nosignature H820IP-7.0.1-1.el6.s390.rpm
```

## Removing H820IP

If H821 is installed it should be removed prior to the removal of H820IP.

During RPM removal, any customized files and log files (e.g., **ntx\_default**, **ntx\_pam**, **prodconf**, **NESikeys**, etc.) will not be deleted. Remove the software as super user with the command:

```
# yum erase H820IP
```

or

```
# rpm -e H820IP
```

## Removing the NESi Public Key

To remove the NESi public key, as super user issue the command:

```
# rpm -e gpg-pubkey-3d6b35d3-51bb5907
```

## Starting, Stopping & Verifying Install of Netex

The service command should be used to stop & start BFX:

```
# service netex stop
# service netex start
# service netex restart
```

The chkconfig command should be used to verify installation:

```
# chkconfig --list netex
```

# Post Installation Considerations

## Configuring H820IP

Once the software package installation has been successfully completed, NetEx/IP must be configured prior to execution. The following instructions address editing the files associated with product activation (`/usr/share/nesi/netex/site/prodconf` and `/usr/share/nesi/NESikeys`), the configuration file for NetEx/IP (`/usr/share/nesi/netex/site/ntx_default`), creating a Network Configuration Table (NCT), creating the Physical Address Map file (PAM file), and starting the NetEx/IP process.

Configuring the H820IP NetEx/IP software consists of the following steps:

- Step 1 Edit the 'NESikeys' file
- Step 2 Configuring the default file
- Step 3 Build an NCT
- Step 4 Create NetEx/IP addressing information
- Step 5 Create Code Conversion Table (optional)
- Step 6 Start NetEx
- Step 7 Verify that 'ntx' Starts Automatically On Reboot
- Step 8 Setup syslog file definition & logrotate rules

### Step 1. Edit the 'NESikeys' file

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

- The default file name is *NESikeys*.
- The `LICPATH` keyword/value pair in the product configuration (see `/usr/share/nesi/netex/site/prodconf`) specifies the full path name to this file. The default is: `/usr/share/nesi/NESikeys`.
- A leading '#', '\*', or ';' character, in a file record denotes a comment line.
- The systems programmer installing this software must edit this file to add a new encrypted Software Key each time such a key is obtained from NESi for H820IP and/or other license-enabled NESi products. This should be done prior to installing the product, and must be done prior to any attempt to run the product successfully.
- To obtain a key from NESi, contact NESi support, supplying the hostname of the machine the software is to be installed on. The hostname may be obtained by issuing the Linux command "*hostname*" with no parameters.
- The file may contain multiple keys per product due to new product releases or a change to the platform's fingerprint (on UNIX this corresponds to the hostname for the target host). If there are multiple keys the NESi product will use the first key found that matches the product name and system fingerprint starting at the top of the file. This makes it important to add new keys for an existing system to the top of the file. For example, if a new key is installed that provides a license date extension, or adds a new feature, adding this new key to the file before the old key ensures the new key will be used rather than the old key. To make the file easier to maintain over time, it is recommended that you precede each Key entry with a

comment line that documents the product designator (e.g., H820IP) and release level of the product that the key is associated with. It will then be easier to delete older Keys that are no longer valid for the product.

- The following shows an example of what a *NEsikeys* file might look like after adding several Keys to the file:

```
# Network Executive Software, Inc. Software License Key file
# Key for H820IP R7.0.1:
CGGZ-4AAA-AAAE-IAO5-O5OJ-SBHX-AUZ5-PL4D
```

## Step 2. Configuring the default file

See Appendix I: NetEx Default File Configuration on page 205.

## Step 3. Build an NCT

You must customize the Network Configuration Table (NCT) for your system and network. As distributed, H820IP does not provide an NCT for your site. You must create the file locally on the host and transfer the file to NetEx/IP in order to access the network configuration.

To generate the NCT, perform the following tasks:

4. Move to the conf directory by entering the following commands:

```
# cd /usr/share/nesi/netex/site
```

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

A sample file (nct.txt.samp) is provided with H820IP in the site directory.

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

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

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

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

```
# cm
Config>
Config> nct nct_conf_file
Config> select *
Config> makepam local_hostname ntx_pam
Config> exit
```

## Step 4. Create NetEx/IP addressing information

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

### 1. Update DNS nameserver information.

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

```
NTX0000uuss
```

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

### 2. Update local host table.(normally only for the local host)

This method requires that you update the local host entry in the local hosts table (usually located at '/etc/hosts') with the IP address and IP hostname (same format as in #1).

### 3. Use the SET IPROUTE command in the ntx\_default file.

The third option is to update the NetEx/IP network information via the new NTXOPER command **SET IPROUTE**. Please refer to the 'Chapter 8: Operator Interface' section of the "*NetEx/IP for UNIX or Windows Systems*" manual for a description of the command. These commands may be placed in the ntx\_default file after the 'ENDINI' statement so that they are executed on every startup.

### 4. Use the SET IPROUTE command

The last option is to update the NetEx/IP network information via the new NTXOPER command **SET IPROUTE**. Please refer to the 'Chapter 8: Operator Interface' section of the "*NetEx/IP for UNIX or Windows Systems*" manual for a description of the command.

**Note:** NESi recommends that customers use options #1 #2 or #3 when updating the IP addressing. Updates made via option #4 are only valid for as long as NetEx/IP is running. IP routing information entered with the 'set iproute' command is lost upon recycle of NetEx/IP or dynamic reload of NCT via the 'load nct' NTXOPER command.

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

These addresses comprise the HYPERChannel *toGNA* addresses, as defined in the NetEx/IP NCT by the NETADDR (*uu*) and SMGDREF (*ss*) parameters.

## Step 5. Create Code Conversion Table (optional)

The code conversion table can now be modified without the need for the product source code or a recompilation of the software.

Customers can execute the command "*cctbld filename*" (where filename is a file location on the host) to create a text file version that they can edit. NetEx will look for the code conversion table at /usr/share/nesi/netex/site/cctable. However, if the customer creates the file in an alternate location, they must add the variable "cctable" to the "NTX\_DEFAULT" file along with the fully qualified pathname of the file.

NetEx/IP checks for the existence of a code conversion file during startup. If an alternate code conversion table does not exist, NetEx/IP will use the built-in default table.

## Step 6. Start NetEx

On an initial install NetEx will not start automatically nor will it start following an update. To Start NetEx use the following command:

```
service netex start
```

## Step 7. Verify that 'ntx' Starts Automatically On Reboot

NetEx has been configured to automatically start for run levels 3, 4, and 5 after a system reboot. It will not work until the 'ntx\_default' and 'ntx\_pam' files have been properly set up and placed in the correct locations, and the key is correct and in the correct location.

**Note:** If NetEx is started/stopped manually, the following script should be used as it properly modifies some system parameters required by NetEx and detects common problems:

```
service netex start | stop | restart | status
```

## Step 8. Setup syslog file definition & logrotate rules

Netex will send most messages to syslog using the facility 'local3' or what is defined in the ntx\_default file. The messages will have a syslog priority that is mapped to a Netex message severity. See "**msgsyslogfac**" for more details.

An example of the syslog definitions for creating netex logs from syslog as defined in **/usr/share/nesi/netex/site/ntx.syslog.cnf** is provided:

```
# rsyslog config for netex
local3.* /var/log/ntxloc3.log
```

The Netex logs may be rotated using your systems log rotation facility. An example of the file **/etc/logrotate.d/ntxlogs.logrotate** is also provided:

```
# netex logrotate config:
# rotate product log files
/usr/share/nesi/netex/site/ntxlog /usr/share/nesi/netex/site/tnp.log
/usr/share/nesi/netex/site/tnpntx.log {
    daily
    compress
    nocreate
    copytruncate
    nodateext
    maxage 365
    rotate 32
    missingok
    #notifempty
    ifempty
    create 644 root root
}

# rotate Netex syslog files
/var/log/ntxloc3.log {
    daily
    compress
    nocreate
    copytruncate
    nodateext
    maxage 365
```

```
rotate 32
missingok
#notifempty
ifempty
create 644 root root
}
```



# Appendix G: H140IP Window on Intel x86 Platforms Installation

## Prerequisites

The following are hardware and software prerequisites for installing the H140IP product.

- An Intel x86 System Windows Server OS (i.e. Windows 2008, Windows 2012, etc).
- At least one other processor on the network running NetEx/IP software. This processor should be connected with another NetEx/IP (not required for intra-host test/evaluation).
- Customers must obtain a software KEY from NESi prior to running the H140IP software. Customers must contact NESi customer support with the customer site name, hostname, and the NetEx/IP product designator (e.g., H140IP). NESi customer support will supply the necessary key once this information has been received. The customer needs to place this key into the NESikeys file as discussed later in this section.

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

## Hardware Installation

Install and verify proper operation of the appropriate operating system.

## Accessing the H140IP software distribution

The H140IP NetEx/IP software is available as a .EXE which may be downloaded from NESi. Contact NESi Customer Support to request the download link.

## Software Installation

The installation is performed by executing the install program with administrator authority. (Once the program is downloaded, right-click and select Run as Administrator)

## Upgrading H140IP

At this time, re-installing or upgrading of this product requires that the current or previous version be removed by running “Add Remove Programs” from the Control Panel.

## Removing H140IP

Removing this product is performed by running “Add Remove Programs” from the Control Panel .

## Starting and Stopping NetexIP

The standard service utility should be used to stop and start NetexIP.

As Administrator, the command line utility “net start NetexIP” or “net stop NetexIP” may also be used.

# Post Installation Considerations

## Configuring H140IP

Once the software package installation has been successfully completed, NetEx/IP must be configured prior to execution. The following instructions address editing the files associated with product activation. These files reside in:

**%ALLUSERSPROFILE%/NESi/**                      **The NESi directory**

and

**%ALLUSERSPROFILE%/NESi/Ntx**                **The Ntx directory**

The NESikeys.ini file is in the NESi directory

The all other files are in the Ntx directory. (prodconf.ini, ntxdefault.ini, ntx\_pam.bin, NCT)

All file locations are relative to the directory that contains the ntx\_default.ini file.

The configuration of NetEx/IP, creating a Network Configuration Table (NCT), and creating the Physical Address Map file (PAM file) follow.

Configuring the H140IP NetEx/IP software consists of the following steps:

Step 1 Edit the 'NESikeys' file

Step 2 Configuring the default file

Step 3 Build an NCT

Step 4 Create NetEx/IP addressing information

Step 5 Create Code Conversion Table (optional) Step 6 Start NetEx

Step 7 Verify that 'ntx' Starts Automatically On Reboot

### Step 1. Edit the 'NESikeys' file

The all other files are in the Ntx directory. (prodconf.ini, ntxdefault.ini, ntx\_pam.bin, NCT)

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

- The default file name is *NESikeys.ini* in the *NESi* directory.
- The LICPATH keyword/value pair in the product configuration file (prodconf.ini in the NESi directory) specifies the full or relative path name to this file. The default is: *..\NESikeys.ini*.
- A leading '#', '\*', or ';' character, in a file record denotes a comment line.
- The systems programmer installing this software must edit this file to add a new encrypted Software Key each time such a key is obtained from NESi for NetexIP and/or other license-enabled NESi products. This must be done prior to any attempt to run the product successfully.
- To obtain a key from NESi, contact NESi support, supplying the hostname of the machine the software is to be installed on. The hostname may be obtained by issuing the command "*hostname*" from a command prompt.
- The file may contain multiple keys per product due to new product releases or a change to the platform's fingerprint (on UNIX this corresponds to the hostname for the target host). If there are multiple keys the

NESi product will use the first key found that matches the product name and system fingerprint starting at the top of the file. This makes it important to add new keys for an existing system to the top of the file. For example, if a new key is installed that provides a license date extension, or adds a new feature, adding this new key to the file before the old key ensures the new key will be used rather than the old key. To make the file easier to maintain over time, it is recommended that you precede each Key entry with a comment line that documents the product designator (e.g., H140IP) and release level of the product that the key is associated with. It will then be easier to delete older Keys that are no longer valid for the product.

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

```
# Network Executive Software, Inc. Software License Key file
# Key for NetexIP H140IP R7.0.2:
CGGZ-4AAA-AAAE-IAO5-O5OJ-SBHX-AUZ5-PL4D
```

## Step 2. Configuring the default file

See Appendix I: NetEx Default File Configuration on page 205.

## Step 3. Build an NCT

You must customize the Network Configuration Table (NCT) for your system and network. As distributed, H140IP provides a sample NCT for your site. You may create the file locally on the host in the Ntx directory in order to access the network configuration.

To generate the NCT, perform the following tasks:

1. Move to the Ntx directory by entering the following commands at a command prompt:

```
# cd %ALLUSERSPROFILE%\NESi\Ntx
```

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

A sample file (nct.txt.samp) is provided with H140IP in the Ntx directory.

2. Invoke the Configuration Manager (CM), as administrator, to process the configuration file by executing CM.

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

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

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

```
C:\> cm
Config>
Config> nct nct_conf_file
```

```
Config> select *
Config> makepam local_hostname ntx_pam.bin
Config> exit
```

## Step 4. Create NetEx/IP addressing information

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

### 1. Update DNS nameserver information.

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

```
NTX0000uuss
```

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

One technique is to create an alias or CNAME for the NTX0000uuss entry referring to the hostname of the target host: ie.

```
NTX00005101 CNAME servername
```

### 2. Use the *SET IPROUTE* command in the *ntx\_default* file.

The second option is to update the NetEx/IP network information via the new NTXOPER command **SET IPROUTE**. Please refer to the ‘Chapter 8: Operator Interface’ section of the “*NetEx/IP for UNIX or Windows Systems*” manual for a description of the command. These commands may be placed in the *ntx\_default* file after the ‘ENDINI’ statement so that they are executed on every startup.

### 3. Use the **SET IPROUTE** command

The last option is to update the NetEx/IP network information via the new NTXOPER command **SET IPROUTE**. Please refer to the ‘Chapter 8: Operator Interface’ section of the “*NetEx/IP for UNIX or Windows Systems*” manual for a description of the command.

**Note:** NESi recommends that customers use options #1 #2 or #3 when updating the IP addressing. Updates made via option #4 are only valid for as long as NetEx/IP is running. IP routing information entered with the ‘set iproute’ command is lost upon recycle of NetEx/IP or dynamic reload of NCT via the ‘load nct’ NTXOPER command.

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

These addresses comprise the HYPERChannel *toGNA* addresses, as defined in the NetEx/IP NCT by the NETADDR (*uu*) and SMGDREF (*ss*) parameters.

## Step 5. Create Code Conversion Table (optional)

The code conversion table can be modified without a recompile.

Customers can execute the command “C:\NESI\netex\cmds\cctbld.exe” to create a text file version that may be edited. The default location for the file is C:\NESI\netex\site\cctable.

If, however, the customer creates the file in an alternate location, the variable “cctable” must be added to NTX\_DEFAULT along with the fully qualified pathname of the file.

NetEx checks for the existence of a code conversion file during startup. If an alternate code conversion table does not exist, NetEx will use the built-in default table.

## **Step 6. Start NetEx**

On an initial install NetEx will not start automatically. To Start NetExIP use either the Services admin tool from the Control Panel or from a CMD prompt as Administrator issue “net start NetexIP”

## **Step 7. Verify that ‘ntx’ Starts Automatically On Reboot**

NetEx has been configured to automatically start after a system reboot. It will not work until the ‘ntx\_default.ini’, ‘NESikeys.ini’, and ‘ntx\_pam.bin’ files have been properly set up.



# Appendix H: H690IP Solaris on Sparc Installation

## Prerequisites

The following are hardware and software prerequisites for installing the H820IP product.

- A Sparc System running Solaris® 9,10 or 11.
- At least one other processor on the network running NetEx/IP software. This processor should be connected with another NetEx/IP (not required for intra-host test/evaluation).
- Customers must obtain a software KEY from NESi prior to running the H690IP software. New keys are required starting with H690IP 7.0.2. Customers must contact NESi customer support with the customer site name, hostname, and the NetEx/IP product designator (e.g., H690IP). NESi customer support will supply the necessary key once this information has been received. The customer needs to place this key into the NESikeys file as discussed later in this section.

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

## Hardware Installation

Install and verify proper operation of the appropriate operating system.

## Accessing the H690IP software distribution

The H690IP NetEx/IP software is available as a Solaris package which may be downloaded from NESi. Contact NESi Customer Support to request the download link.

## Upgrading H690IP

If you are upgrading an existing installation of H690IP, it is strongly recommended that any running NetEx/IP process be stopped. Copy the ntx\_default, ntx\_pam, prodconf, NESikeys and any special code conversion table files to a temporary location.

If you are upgrading from a package older than H690IP 7.0.2, the package should be removed using the pkgrm command. Additionally if you have NetEx startup procedures in the boot process you will need to remove it (i.e. inittab, rc scripts, SMF, etc.) Once these are done, you can proceed to the Software Installation.

To find the version of the installed package use the following command:

```
# pkginfo -l NESiNTX
```

## Removing H690IP

If H691 is installed it should be removed prior to the removal of H690IP.

Insure you save the NESikeys file prior to removal if you are running other NetEx products which require a software key.

If you are planning on re-installing H690IP, copy the ntx\_default, PAM, NCT, and modified code conversion tables and modified prodconf files to a temporary location.

As super user, remove the software with the pkgrm command:

```
# pkgrm NESiNTX
```

## Starting, Stopping & Verifying Install of Netex/IP

If you are running an OS version older than Solaris 10, you will be using init script process for starting, stopping and status of NetEx/IP.

```
# /etc/init.d/netex stop
# /etc/init.d/netex start
# /etc/init.d/netex restart
# /etc/init.d/netex status
```

If you are using SMF, you will be using the following commands to start, stop and retrieve status of NetEx/IP:

```
# svcadm disable netex
# svcadm enable netex
# svcadm restart netex
# svcs -l netex
```

*Note: Upon installation, you will need to enable NetEx/IP after installation of keys and configuration of NetEx/IP.*

## Software Installation

Please review all the installation instructions prior to performing the install.

If this is an initial installation, install the software as super user with the command:

```
# uncompress ./H690IP-7.0.2.pkg.Z
# pkgadd -d ./H690IP-7.0.2.pkg
```



## Configuring H690IP

Once the software package installation has been successfully completed, NetEx/IP must be configured prior to execution. The following instructions address editing the files associated with product activation (`/usr/share/nesi/netex/site/prodconf` and `/usr/share/nesi/NESikeys`), the configuration file for NetEx/IP (`/usr/share/nesi/netex/site/ntx_default`), creating a Network Configuration Table (NCT), creating the Physical Address Map file (PAM file), and starting the NetEx/IP process.

Configuring the H690IP NetEx/IP software consists of the following steps:

- Step 1 Edit the 'NESikeys' file
- Step 2 Configuring the default file
- Step 3 Build an NCT
- Step 4 Create NetEx/IP addressing information
- Step 5 Create Code Conversion Table (optional)
- Step 6 Start NetEx
- Step 7 Verify that 'ntx' Starts Automatically On Reboot
- Step 8 Setup syslog file definition & logrotate rules

### Step 1. Edit the 'NESikeys' file

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

- The default file name is *NESikeys*.
- The `LICPATH` keyword/value pair in the product configuration file (see `/usr/share/nesi/netex/site/prodconf`) specifies the full path name to this file. The default is: `/usr/share/nesi/NESikeys`.
- A leading '#', '\*', or ';' character, in a file record denotes a comment line.
- The systems programmer installing this software must edit this file to add a new encrypted Software Key each time such a key is obtained from NESi for H820IP and/or other license-enabled NESi products. This should be done prior to installing the product, and must be done prior to any attempt to run the product successfully.
- To obtain a key from NESi, contact NESi support, supplying the hostname of the machine the software is to be installed on. The hostname may be obtained by issuing the Linux command "`hostname`" with no parameters.
- The file may contain multiple keys per product due to new product releases or a change to the platform's fingerprint (on UNIX this corresponds to the hostname for the target host). If there are multiple keys the NESi product will use the first key found that matches the product name and system fingerprint starting at the top of the file. This makes it important to add new keys for an existing system to the top of the file. For example, if a new key is installed that provides a license date extension, or adds a new feature, adding this new key to the file before the old key ensures the new key will be used rather than the old key. To make the file easier to maintain over time, it is recommended that you precede each Key entry with a comment line that documents the product designator (e.g., H690IP) and release level of the product that

the key is associated with. It will then be easier to delete older Keys that are no longer valid for the product.

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

```
# Network Executive Software, Inc. Software License Key file
# Key for H690IP R7.0.2:
CGGZ-4AAA-AAAE-IAO5-O5OJ-SBHX-AUZ5-PL4D
```

## Step 2. Configuring the default file

If upgrading, merge your specific site settings into the `ntx_default` file which was installed with this release. See Appendix I: NetEx Default File Configuration on page 205.

## Step 3. Build an NCT

If upgrading, copy your saved NCT and your PAM to `/usr/share/nesi/netex/site`, and then skip to Step 4.

Otherwise, you must customize the Network Configuration Table (NCT) for your system and network. As distributed, H690IP does not provide an NCT for your site. You must create the file locally on the host and transfer the file to NetEx/IP in order to access the network configuration.

To generate the NCT, perform the following tasks:

1. Move to the site directory by entering the following commands:

```
# cd /usr/share/nesi/netex/site
```

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

A sample file (`nct.txt.samp`) is provided with H690IP in the site directory.

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

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

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

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

```
# cm
Config>
Config> nct nct_conf_file
Config> select *
Config> makepam local_hostname ntx_pam
Config> exit
```

## Step 4. Create NetEx/IP addressing information

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

### 1. Update DNS nameserver information.

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

```
NTX0000uu $\mathit{ss}$ 
```

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

### 2. Update local host table.(normally only for the local host)

This method requires that you update the local host entry in the local hosts table (usually located at '/etc/hosts') with the IP address and IP hostname (same format as in #1).

### 3. Use the SET IPRROUTE command in the ntx\_default file.

The third option is to update the NetEx/IP network information via the new NTXOPER command **SET IPRROUTE**. Please refer to the 'Chapter 8: Operator Interface' section of the "*NetEx/IP for UNIX or Windows Systems*" manual for a description of the command. These commands may be placed in the ntx\_default file after the 'ENDINI' statement so that they are executed on every startup.

### 4. Use the SET IPRROUTE command

The last option is to update the NetEx/IP network information via the new NTXOPER command **SET IPRROUTE**. Please refer to the 'Chapter 8: Operator Interface' section of the "*NetEx/IP for UNIX or Windows Systems*" manual for a description of the command.

**Note:** NESi recommends that customers use options #1 #2 or #3 when updating the IP addressing. Updates made via option #4 are only valid for as long as NetEx/IP is running. IP routing information entered with the 'set iproute' command is lost upon recycle of NetEx/IP or dynamic reload of NCT via the 'load nct' NTXOPER command.

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

These addresses comprise the HYPERChannel *toGNA* addresses, as defined in the NetEx/IP NCT by the NETADDR (*uu*) and SMGDREF (*ss*) parameters.

## Step 5. Create Code Conversion Table (optional)

If upgrading and you have a saved code conversion table, copy it to /usr/share/nesi/netex/site, and go on to Step 6 Start NetEx.

The code conversion table can now be modified without the need for the product source code or a recompilation of the software.

Customers can execute the command "*cctbld filename*" (where filename is a file location on the host) to create a text file version that they can edit. NetEx will look for the code conversion table at /usr/share/nesi/netex/site/cctable. However, if the customer creates the file in an alternate location, they must add the variable "cctable" to the "NTX\_DEFAULT" file along with the fully qualified pathname of the file.

NetEx/IP checks for the existence of a code conversion file during startup. If an alternate code conversion table does not exist, NetEx/IP will use the built-in default table.

## Step 6. Start NetEx

On an initial install NetEx will not start automatically. Refer to the section on page 200 to Start NetEx.

## Step 7. Verify that 'ntx' Starts Automatically On Reboot

NetEx has been configured to automatically start after a system reboot. It will not work until the 'ntx\_default' and 'ntx\_pam' files have been properly set up and placed in the correct locations, and the key is correct and in the correct location.

## Step 8. Setup syslog file definition & logrotate rules

Netex will send most messages to syslog using the facility 'local3' or what is defined in the ntx\_default file. The messages will have a syslog priority that is mapped to a Netex message severity. See "msgsyslogfac" for more details.

An example of the syslog definitions for creating netex logs from syslog as defined in /usr/share/nesi/netex/site/ntx.syslog.cnf is provided:

```
# rsyslog config for netex
local3.* /var/log/ntxloc3.log
```

The Netex logs may be rotated using your systems log rotation facility. An example of the file /etc/logrotate.d/ntxlogs.logrotate is also provided:

```
# netex logrotate config:
# rotate product log files
/usr/share/nesi/netex/site/ntxlog /usr/share/nesi/netex/site/tnp.log
/usr/share/nesi/netex/site/tnpntx.log {
    daily
    compress
    nocreate
    copytruncate
    nodateext
    maxage 365
    rotate 32
    missingok
    #notifempty
    ifempty
    create 644 root root
}

# rotate Netex syslog files
/var/log/ntxloc3.log {
    daily
    compress
    nocreate
    copytruncate
    nodateext
    maxage 365
    rotate 32
    missingok
    #notifempty
    ifempty
    create 644 root root
}
```

# Appendix I: NetEx Default File Configuration

## Edit the NTX\_DEFAULT file

The *ntx\_default* file contains default values for NetEx/IP parameters.

*Note for UNIX only: After an update, if a *ntx\_default.rpmnew* exists there may be updated defaults that should be reviewed.*

Edit this file with the following recommendations:

1. There are two entries that must be supplied by the customer prior to starting NETEX: local and device(x). NETEX will not run if these are commented out of the *ntx\_default* file.
2. The first entry (local) in the file defines the name of your local host. Edit this entry to reflect the name of your system as it appears in the Network Configuration Table (NCT). The “device1, device2, etc entries in the table below (‘UDP-6950-0000UUSS’) are not the default values for those particular parameters. The user will need to add these parameters to their “*ntx\_default*” file where *UUSS* is the NETADDR and SMGDREF specified in the NCT for the IP interface. The ‘6950’ is the UDP port. Edit or modify any other parameters for your host and site. The following table lists all of the parameters and their default values.

Lines preceded by an ‘\*’, ‘#’, ‘”’, ‘!’ or ‘/’ are comments and indicate the use of provided program defaults. To override these default values you should duplicate the entry and uncomment, remove the \*, and supply your override value. Note that parameters for Protocol Type 4 should not be altered unless directed by Netex Support, as this protocol is currently not supported.

NTX_Parameter	Default	Definition
local	NETEX	This is the NetEx name of your host. The local host name is the same name defined on the HOST statement in the NCT and during the MAKEPAM phase when building the PAM file.
logerrors	default	The two allowed values are “default” and “all”: <b>DEFAULT:</b> This value logs adapter and communications errors that are uncommon. Some of these errors are normal but can occur only once every few hours. <b>ALL:</b> This value logs all adapter and communications errors.

NTX_Parameter	Default	Definition
mbxname	H800IP, H320IP only: /var/opt/netex/netexserver H820IP, H370IP only: /usr/share/nesi/netex/netexserver H690IP, H620IP only: /usr/nesi/netex/netexserver	UNIX only This is the name of the pipe used to move NRB and DATA requests from the user address space into NetEx/IP address space. This name is built into the user interface routines. If this is changed, the code in the interface routines must be changed and the interface routines rebuilt. If this is changed, the environmental variable NETEX_DEVICE must be set to the full path name of the new pipe in each job that references NetEx/IP.
pamfile	(Unix) /usr/share/nesi/netex/site/ ntx_pam (Windows) %ALLUSERSPROFILE %/NESi\Ntx\ntx_pam.bin	This is the name of the PAM file NetEx/IP will use to load the initial NCT. This name should match the name specified in the Configuration Manager MAKEPAM command.
ntxlogname	(Unix) /usr/share/nesi/netex/site/ ntxlog (Windows) %ALLUSERSPROFILE %/NESi\Ntx\ntx.log	This is the name of the NetEx/IP log file. Messages associated with the operation of NetEx/IP are logged in this file. If this is changed, then the logrotate and syslog control files must also be changed.
cctable	(Unix) /usr/share/nesi/netex/site/ cctable (Windows) %ALLUSERSPROFILE %/NESi\Ntx\cctable.txt	This is the name of a site modified NetEx code conversion table.
msgsyslog	3	0: send minimal output to the Netex log and no output to syslog 1: send Netex msgs to Netex log. 2 : send Netex msgs to system log. 3 : send them to both logs.
msgsyslogfac	local3	See SET MSGSYSLOGFAC
trapcmd	/usr/share/nesi/netex/scripts/netex_trap.sh	Not supported at this time
ackcredit	2	This is the number of buffers that NetEx/IP sends without returning an explicit ack. <b>Note:</b> More buffers may be in the process of being sent. Increasing this value has little effect on system load and may decrease NetEx/IP throughput.

NTX_Parameter	Default	Definition
bufcnt	2000	This is the number of “segment” sized buffers available for all input and output. <b>Note:</b> This parameter should be increased to match the maximum amount of data expected to be in transit (being sent but not yet acknowledged) divided by “segment”.
commipod	300	Percentage factor of deadtime used to determine communications interrupted time.
connto	30	This is the initial value for CONNECT TIMEOUT
datato	30	This is the initial value for DATA TIMEOUT. In a heavily loaded system or during transfers to tape, this value should be increased to about 1000.
deadto	60	This is the initial value for DEAD TIMEOUT.
defblkkin	32768	This is the default value for NRBBLKIN, if the value zero is specified.
defblkout	32768	This is the default value for NRBBLKOUT, if the value zero is specified.
defmsecsonewaydelay	0	This is the one-way propagation delay of the network, expressed in milliseconds, and is used if there is no delay value specified in the PAM for this network path.  For Type 2 protocol connections, this value represents a fixed propagation delay that never changes.  For Type 4 protocol connections, this value represents a starting point that is used for internal bandwidth capacity calculations. However, the delay is continuously measured during each session, and if it changes, the updated value is used for subsequent internal bandwidth capacity calculations.
dreadqueue	12	This is the number of reads that NetEx/IP has outstanding to the DRIVER. This value may need to be increased, if many very small transfers are needed.
dynpam	0	0 - netex is not using dynamic pams (NCT is necessary)  1 - netex is using dynamic pams (NCT not necessary - only supported with HyperIP)
highresmsecs	10	This is the value of the internal high resolution timer. It is used for internal rate throttling, and should not be changed.

<b>NTX_Parameter</b>	<b>Default</b>	<b>Definition</b>
idleto	6	This is the initial value for IDLE TIME. In networks with many errors or contention, it may help to drop this value to 2 or 3.
maxblkkin	65400	This is the maximum NRBBLKIN value in bytes.
maxblkout	65400	This is the maximum NRBBLKOUT value in bytes.
maxkbitspersec	0	This is the maximum rate at which NetEx/IP will deliver data to the network for each network connection. If zero is specified, data will be delivered with no internal throttling. This value is only used for connections to hosts that do not have a 'rate' value specified in the PAM. This value is specified in Kbits per second.  For example, a value of 50 means 50Kbs; a value of 50000 means 50,000 Kbs (i.e., 50Mbs).
maxmbxxfer	32768	This is the maximum block that can be sent across the STREAMS pipe facility.
mbufin	5	This is the number of input blocks that NetEx/IP will allow to be outstanding for each user. This may have to be increased for long telecommunications delays.
mbufout	5	This is the number of output blocks that NetEx/IP will allow to be outstanding for each user. This may have to be increased for long telecommunications delays.
msglvl	interesting	This is the value that controls the verbosity of the message displays. A value must be specified as one of the following:  <ul style="list-style-type: none"> <li>immediate</li> <li>important</li> <li>interesting</li> <li>moderate</li> <li>monitor</li> <li>debug</li> <li>blither</li> </ul> A value of 'important' (or lower) must be specified to enable NetEx/IP session establishment messages to be recorded in the 'ntxlog' file.



<b>NTX_Parameter</b>	<b>Default</b>	<b>Definition</b>
multihost	Off	Specifies whether multihost is enabled or disabled. This parameter is important when using the TNP feature.
nodns	0	<p>Enable or disable DNS lookups when the PAM file is loaded. Accepted values are “1” and “0”. The default is 0 (false).</p> <p>When set to <b>1</b> (true), NetEx/IP will skip DNS lookups when the PAM file is read.</p> <p>This is only useful if the user intends to <b>ONLY</b> use the “SET IPROUTE” command to manually map “toGNA” addresses to IP addresses.</p>
numlogs	5	This value defines the maximum number of ‘ntxlog’ files that are saved by the ‘swlog’ command. (swlog is not available in all distributions)
pollsel	On	This specifies whether or not the dispatcher will poll with 0 timeout or not. Especially useful in order to get good performance with small segments.
prefprot	2	This value defines the default preferred protocol type to use when connecting to hosts that support multiple NetEx/IP protocol types. Valid values are 2 or 4.
rcvratesecs	2	This is the time interval (in seconds) after which the receive rate for a network connection is recalculated. The receive rate refers to the rate at which the application is receiving data from NetEx.
ropclass	G	<p>This defines the default remote operator class for the NetEx/IP operator console. Allowed values are A or G. The default is “A”.</p> <p>“A” will allow privileged instructions such as SET, HALT, etc. to be issued via the remote operator facility.</p> <p>“G” only allows non-privileged operator commands (DISPLAY, etc.).</p>
segsize	32768	This is the maximum size of a NetEx data block on the network. .
slim	255	This is the limit on the smax parameter.
smax	32	This is the maximum number of sessions active at any one time. If this value is changed, the file specified by “mbxname” above should be deleted.
smqmax	25	Dictates the maximum session manager request queue depth

<b>NTX_Parameter</b>	<b>Default</b>	<b>Definition</b>
timer	2	This is the initial value for the watchdog timer.
debugdata	0	This is the maximum number of data bytes to trace for any associated data blocks.
debugmsg	0	This parameter enables or disables the tracing of HYPERchannel messages between NetEx/IP and the network. A value of 0 turns tracing off; any other value turns tracing on.
debugreq	0	This parameter enables or disables the tracing of user requests arriving at the NetEx/IP protocol stack. A value of 0 turns tracing off; any other value turns tracing on. The default is 0.
debugret	0	This parameter enables or disables the tracing of user responses being returned from the NetEx/IP protocol stack. A value of 0 turns tracing off; any other value turns tracing on. The default is 0.
udpport	6950	Only used with nctless NetEx (not supported at this time), otherwise see device statements
iprecv	250000	IP Receive buffer size
ipsend	250000	IP Send buffer size
ipchksum	1	Not implemented
ip1intrf	1	If true (1), one IP socket/device for all interfaces. If false (0), a socket/device for each IP interface.
device1	UDP-6950-0000UUSS	This is the device name of the IP device interface. UUSS is the NETADDR and SMGDREF specified in the NCT for the IP interface. The '6950' is the UDP port.
device2	(blank – this indicates other interfaces do not exist)	This is the device name of the second network device interface.
device3	(blank – this indicates other interfaces do not exist)	This is the device name of the third network device interface.
device4	(blank – this indicates other interfaces do not exist)	This is the device name of the fourth network device interface.
The following parameters are only applicable to protocol type 4 connections (and are currently not supported):		
bufolimit	2000	This is the maximum number of outstanding buffers that Netex will allow per connection.

<b>NTX_Parameter</b>	<b>Default</b>	<b>Definition</b>
defstartkbitspersec	455000	<p>This is the starting throughput rate at which NetEx/IP will attempt to deliver data to the network for each network connection. If zero is specified, data will be delivered with no internal throttling. This value is only used for connections to hosts that do not have a 'rate' value specified in the PAM or NRB.</p> <p>This value is specified in Kbits per second. For example, a value of 50 means 50Kbs; a value of 50000 means 50Mbs (i.e. 50,000 Kbs).</p>
delincp	900	Percentage factor used when decreasing speed because of delay increase.
delincsreq	750	Percentage factor of send rate used when determining equivalency of increase or decrease of delay.
delincrreq	750	Percentage factor of receive rate used when determining equivalency of increase or decrease of delay.
minkbitspersec	0	This is the minimum throughput rate that Netex will decrease to.
minnpdu	1300	Currently not supported.
segdownp	300	Currently not supported.
segupp	200	Currently not supported.
startnpdu	6500	Currently not supported.
modsegsz	0	This indicates whether Netex may modify the segment size dynamically. This should remain at 0 currently and is not supported.
oktodec	1	This indicates whether Netex is allowed to decrease speed or not.
nakdec	0	This indicates whether Netex will decrease speed when there are NAKs occurring.
ratedelaydecp	250 (25.0%)	This is the percentage factor used to decrease the sending rate of a network connection if the round-trip delay increases. This recalculation is performed after the expiration of each interval specified by the 'rtdelayincsecs' parameter. The value specified represents a percentage multiplied by a factor of 10.
rateequivph	855 (85.5%)	This is the high bound of the send and receive equivalence adjustment (see 'rateequivps'). The value specified represents a percentage multiplied by a factor of 10.

<b>NTX_Parameter</b>	<b>Default</b>	<b>Definition</b>
rateequivpl	500 (50.0%)	This is the low bound of the send and receive equivalence adjustment (see 'rateequivps'). The value specified represents a percentage multiplied by a factor of 10.
rateequivps	750 (75.0%)	This is the percentage factor used to determine equivalence of the send and receive rates for each network connection. These rates are assumed to be equal if they fall within this percentage of each other. This is the initial value used for each network connection, and can be dynamically adjusted based on activity and performance of the network. The value specified represents a percentage multiplied by a factor of 10.
rcvdataqbytes	20000000	This is the high threshold value (in bytes) for the size of the receiving NetEx DataQue for each network connection. If the size of the Data Queue exceeds this value, subsequent blocks received on the network that are greater than the highest received LRN are NAK'ed and discarded. Subsequent blocks will continue to be discarded until the size of the DataQue is reduced to the value specified by 'rcvdataqlbytes'.
rcvdataqhsegs	15000	This is the high threshold value (in segments) for the size of the receiving NetEx DataQue for each network connection. If the size of the Data Queue exceeds this value, subsequent blocks received on the network that are greater than the highest received LRN are NAK'ed and discarded. Subsequent blocks will continue to be discarded until the size of the DataQue is reduced to the value specified by 'rcvdataqlsegs'.
rcvdataqlbytes	10000000	This is the low threshold value (in bytes) for the size of the receiving NetEx DataQue for each network connection. See the 'rcvdataqbytes' parameter for the description of how this value is used.
rcvdataqlsegs	8000	This is the low threshold value (in segments) for the size of the receiving NetEx DataQue for each network connection. See the 'rcvdataqhsegs' parameter for the description of how this value is used.
rtdelayincsecs	60	This is the interval (in seconds) used by the sending side of a network connection after which a check is made for an increase in the round trip delay.

<b>NTX_Parameter</b>	<b>Default</b>	<b>Definition</b>
rexmwblks	2	This is the number of blocks to use to calculate the time to wait before retransmitting a block when it is NAKed. Only used if userexmitq is 1.
sndrateupsecs	20	This is the interval (in seconds) used by the sending side of a network connection after which a check is made for a rate increase or decrease (i.e. the receive rate is greater or less than 'rateequivps' of the send rate). If an adjustment is required, the sending rate is increased or decreased by the current value of 'sndrateps'.
sndrateph	500 (50.0%)	This is the high bound of the 'sndrateps' adjustment for each network connection. The value specified represents a percentage multiplied by a factor of 10.
sndratepl	100 (10.0%)	This is the low bound of the 'sndrateps' adjustment for each network connection. The value specified represents a percentage multiplied by a factor of 10.
sndrateps	200 (20.0%)	This is the percentage factor used to increase or decrease the send rate for each network connection. The send rate may be increased after the expiration of the interval specified by 'sndrateincsecs'. The send rate may be decreased after the expiration of the interval specified by 'sndratedecsecs'. The value specified represents a percentage multiplied by a factor of 10.
sndratesecs	4	The actual send rate calculation interval
startratep	750 (75.0%)	This is the percentage factor used to calculate the initial send rate of each network connection. This value is applied against the maximum rate for the connection, as specified in the PAM, NRB, or by the 'defstartkbitspersec' parameter. During the course of the connection, the actual send rate may be adjusted, based on network activity. The value specified represents a percentage multiplied by a factor of 10.
usercvgapq	0	This indicates whether or not Netex should use the receive gap queue. This should currently remain at 0.
userexmitq	1	Specifies whether or not to use the retransmit queue.
writeto	45	the number of seconds a write will wait to be accepted by netex before failing

NTX_Parameter	Default	Definition
xdbg	0	This is a special debug option to see details of throttling. Setting the value > 0 will turn on xdbg tracing, default is 0.
ENDINI		Indicates the end of parameters. Anything after this line will be processed as an NTXOPER command.

**Notes:**

- Some of the parameters documented above may not be included in the sample ntx\_default file (“/usr/share/nesi/netex/site/ntx\_default”). It is the responsibility of the user to enter these values as necessary into the installation-specific copy of “ntx\_default” prior to starting NetEx.
- After a re-install or upgrade install, it is possible to have a newer ntx\_default file. If so, it will be named ntx\_default.rpmnew. It is the responsibility of the user to merge or update ntx\_default to reflect any additions or deletions.

# Appendix J: NetEx Default Parameters Mapping

This section maps the NetEx default parameter names with the operator commands and display names. Shaded entries are for NetEx protocol 4 only and are not supported at this time.

<b>ntx_default</b>	<b>initialization</b>	<b>ntxoper display</b>	<b>ntxoper set</b>	<b>negotiated/ calculated</b>
local	local	Host		
*logerrors	logerrors	(not used)		
*mbxname	mbxname			
*pamfile	pamfile		(load nct changes it)	
*ntxlogname	ntxlogname			
*cctable	cctable			
*msgsyslog	msgsyslog	msgsyslog (d p)	msgsyslog	
*msgsyslogfac	msgsyslogfac	msgsyslogfac	msgsyslogfac	
*ropclass	ropclass	class (d p)	ropclass	
*nodns	nodns			
*ackcredit	ackcredit	Ackcr (d t n)		
*bufcnt	bufcnt			
*connto	connto	contime (d p)	contime	
*datato	datato	readtime (d p)	readtime	
*deadto	deadto	deadtime (d p)	deadtime	
*defblkkin	defblkkin	defbi (d p)	defbi	
*defblkout	defblkout	defbo (d p)	defbo	
*dreadqueue	dreadqueue	dreadque (d p)		
*idleto	idleto	idletime (d p)	idletime	
*maxblkkin	maxblkkin	maxbi (d p)	maxbi	Mblki (d t n) - negotiated
*maxblkout	maxblkout	maxbo (d p)	maxbo	Mblko (d t n) - negotiated
*maxmbxxfer	maxmbxxfer			
*mbufin	mbufin	mbufin (d p)	mbufin	Maxrblok (d t n) - calculated

<b>ntx_default</b>	<b>initialization</b>	<b>ntxoper display</b>	<b>ntxoper set</b>	<b>negotiated/ calculated</b>
*mbufout	mbufout	mbufout (d p)	mbufout	Maxtblok (d t n) - calculated
*smqmax	smqmax			
*segsz	segsz	segsz (d p)		Segsz (d t n) - negotiated
*slim	slim	lim ses (d p)		
*smax	smax	max ses (d p)	sesmax	
*timer	timer	wdogint (d p)	wdogint	
*numlogs	numlogs			
*prefprot	prefprot	prefprot (d p)	prefprot	Prot (d t n) - negotiated
*dynpam	dynpam			
*msglvl	msglvl	msglvl (d p)	msglvl	
*multihost	multihost	multihost (d p)	multihost	
*pollsel	pollsel	pollsel (d p)	pollsel	
*userexmitq	userexmitq	userexmitq (d p)	userexmitq	UseReXmQ (d t n)
*rexmwbkls	rexmwbkls	rexmwbkls (d p)	rexmwbkls	ReXmWBlks (d t n)
*usercvgapq	usercvgapq	usercvgapq (d p)	usercvgapq	UseRcvGapQ (d t n)
*writeto	writeto			
*xdbg	xdbg	xdbg (d p)	xdbg	
*maxkbitspersec	maxkbitspersec	mxkbs (d p)	maxkbs	Mrate (d t n) - calculated
*highresmsecs	highresmsecs			
*rcvratesecs	rcvratesecs			
*sndratesecs	sndratesecs			
*defmsecsoneway dela	defmsecsoneway waydelay			
*rcvdataqlbytes	rcvdataqlbytes	dtql (d p)	rcvdataqlb	DtQLB (d t n)
*rcvdataqlsegs	rcvdataqlsegs	dtqls (d p)	rcvdataqls	DtQLS (d t n)
*rcvdataqhbytes	rcvdataqhbytes	dtqh (d p)	rcvdataqhb	DtQHB (d t n)
*rcvdataqhsegs	rcvdataqhsegs	dtqhs (d p)	rcvdataqhs	DtQHS (d t n)



<b>ntx_default</b>	<b>initialization</b>	<b>ntxoper display</b>	<b>ntxoper set</b>	<b>negotiated/ calculated</b>
*startratep	startratep			
*defstartkbitspersec	defstartkbitspersec			
*rtdelayincsecs	rtdelayincsecs			
*ratedelaydec	ratedelaydec			
*sndrateupsecs	sndrateupsecs			
*sndratepl	sndratepl			
*sndrateps	sndrateps			
*sndrateph	sndrateph			
*rateequivpl	rateequivpl			
*rateequivps	rateequivps			
*rateequivph	rateequivph			
*minkbitspersec	minkbitspersec			
*bufolimit	bufolimit	bufolim (d p)	bufolim	BufOLim (d t n)
*oktodec	oktodec			
*nakdec	nakdec			
*modsegsz	modsegsz			
*minnpdu	minnpdu			
*startnpdu	startnpdu			
*segdownp	segdownp			
*segupp	segupp			
*delincp	delincp			
*delincsreq	delincsreq			
*delincrreq	delincrreq			
*commipod	commipod			
*userexmitq	userexmitq	userexmitq (d p)	userexmitq	UseReXmQ (d t n)
*rexmwbks	rexmwbks	rexmwbks (d p)	rexmwbks	ReXmWBks (d t n)
*usercvgap	usercvgapq	usercvgapq (d p)	usercvgapq	UseRcvGapQ (d t n)
*debugreq	debugreq	dbgreq (d p)	dbgreq	

<b>ntx_default</b>	<b>initialization</b>	<b>ntxoper display</b>	<b>ntxoper set</b>	<b>negotiated/ calculated</b>
*debugret	debugret	dbgret (d p)	dbgret	
*debugmsg	debugmsg	dbgmsg (d p)	dbgmsg	
*debugdata	debugdata	dbgdata (d p)	dbgdata	
*udpport	udpport			
*iprecv	iprecv			
*ipsend	ipsend			
*mtudisc	mtudisc	mtudisc (d p)		
*ipchksum	ipchksum			
*iplintrf	iplintrf			
device1	device1			
device2	device2			
device3	device3			
device4	device4			
*usergate	usergate			

# Index

abnormal termination.....	21
alternate path retry (APR).....	2
ASCII.....	vii
asynchronous.....	vii
auto datamode.....	30
block segmenting.....	2
buffer.....	vii
C function	
SCLOS.....	49
SCONF.....	43
SCONN.....	41
SDISC.....	54
SOFFR.....	39
SREAD.....	45
SWAIT.....	51
SWRIT.....	47
C language.....	1
C program example.....	56
C programming interface.....	37
calling programs.....	1
characteristics of NetEx/IP.....	1
code conversion.....	vii, 24
common recovery procedures.....	23
concurrent data transfer.....	16
configuration manager.....	vii
Coprocessor NETwork EXecutive.....	<i>See</i> CP NetEx
design of NetEx/IP.....	2
driver sublayer services.....	7
enabling remote operator.....	71
error codes.....	23
error recovery.....	23
establishing a session.....	12
external interface.....	1
general session concept.....	10
handling multiple connections.....	21
header.....	vii
host.....	vii
I/O flow.....	3
internal operation.....	1
Internet Protocol (IP).....	vii
ISO.....	vii
ISO model.....	5
driver sublayer.....	7
network layer.....	7
session layer.....	6
transport layer.....	6
link.....	vii
manual datamode.....	29
NESiGate Offload NetEx/IP.....	3
NETEX operator (NTXOPER)	
DISPLAY USAGE.....	<i>See</i> DISPLAY MEMORY
NETEX operator (NTXOPER).....	69
CLEAR IPRROUTE.....	75
CLEAR LOG.....	74
command description.....	72
command line mode.....	70
DISPLAY DRAINED.....	76
DISPLAY HOST.....	77
DISPLAY IPRROUTE.....	79
DISPLAY KEY.....	81
DISPLAY LOG.....	82
DISPLAY MEMORY.....	83
DISPLAY NETWORK.....	85
DISPLAY PARMS.....	87
DISPLAY SESSION.....	93
DISPLAY TRANSPORT.....	95
executing commands.....	70
interactive mode.....	70
remote operator service.....	71
NETEX operator (NTXOPER)	
DISPLAY VERSION.....	102
NETEX operator (NTXOPER)	
DRAIN ADAPTER.....	102
NETEX operator (NTXOPER)	
DRAIN NETEX.....	103
NETEX operator (NTXOPER)	
DRAIN HOST.....	104
NETEX operator (NTXOPER)	
HALT SREF.....	105
NETEX operator (NTXOPER)	
HELP.....	106
NETEX operator (NTXOPER)	
KILL NETEX.....	107
NETEX operator (NTXOPER)	
LOAD KEY.....	108
NETEX operator (NTXOPER)	
LOAD NCT.....	109
NETEX operator (NTXOPER)	
SET BUFOLM.....	110
NETEX operator (NTXOPER)	
SET CONTIME.....	111
NETEX operator (NTXOPER)	
SET DBGDATA.....	112
NETEX operator (NTXOPER)	
SET DBMSG.....	113
NETEX operator (NTXOPER)	
SET DBGREQ.....	114
NETEX operator (NTXOPER)	
SET DBGRET.....	115
NETEX operator (NTXOPER)	

SET DEADTIME.....	116
NETEX operator (NTXOPER)	
SET DEFBI.....	117
NETEX operator (NTXOPER)	
SET DEFBO.....	118
NETEX operator (NTXOPER)	
SET IDLETIME.....	119
NETEX operator (NTXOPER)	
SET IROUTE.....	120
NETEX operator (NTXOPER)	
SET MAXBI.....	121
NETEX operator (NTXOPER)	
SET MAXBO.....	122
NETEX operator (NTXOPER)	
SET MAXKBS.....	123
NETEX operator (NTXOPER)	
SET MBUFIN.....	124
NETEX operator (NTXOPER)	
SET MBUFOUT.....	125
NETEX operator (NTXOPER)	
SET MSGLVL.....	126
NETEX operator (NTXOPER)	
SET MSGSYSLOG.....	127
NETEX operator (NTXOPER)	
SET MSGSYSLOGFAC.....	128
NETEX operator (NTXOPER)	
SET MULTIHOST.....	130
NETEX operator (NTXOPER)	
SET NTXOPER.....	131
NETEX operator (NTXOPER)	
SET POLLSEL.....	132
NETEX operator (NTXOPER)	
SET PREFPROT.....	133
NETEX operator (NTXOPER)	
SET RCVDATAQHB.....	134
NETEX operator (NTXOPER)	
SET RCVDATAQLB.....	135
NETEX operator (NTXOPER)	
SET RCVDATAQHS.....	136
NETEX operator (NTXOPER)	
SET RCVDATAQLS.....	137
NETEX operator (NTXOPER)	
SET REXMWBLKS.....	138
NETEX operator (NTXOPER)	
SET READTIME.....	139
NETEX operator (NTXOPER)	
SET ROPCLASS.....	140
NETEX operator (NTXOPER)	
SET SESMAX.....	141
NETEX operator (NTXOPER)	
SET USERCVGAPQ.....	142
NETEX operator (NTXOPER)	
SET USEREXMITQ.....	143
NETEX operator (NTXOPER)	
SET WDOGINT.....	144
NETEX operator (NTXOPER)	

START NETEX.....	144
NETEX operator (NTXOPER)	
START ADAPTER.....	145
NETEX operator (NTXOPER)	
START HOST.....	146
NetEx request block (NRB)	
creation.....	34
duplication.....	34
fields.....	25
NRBBLKI.....	32
NRBBLKO.....	32
NRBBUFA.....	29
NRBBUFL.....	29
NRBCLASS.....	32
NRBDMODE.....	29
NRBIND.....	27
NRBLEN.....	27
NRBMAXRT.....	32
NRBNREF.....	29
NRBOFFER.....	34
NRBOSD.....	34
NRBPROTA.....	33
NRBPROTL.....	33
NRBREQ.....	28
NRBRESV1.....	34
NRBRESV2.....	34
NRBRESV3.....	34
NRBSTAT.....	26
NRBTIME.....	31
NRBUBIT.....	27
NRBUSER.....	34
usage rules.....	25
NETEX request block (NRB)	25
in C.....	37
NRBHOST.....	34
NetEx/IP characteristics.....	1
NetEx/IP connections.....	1
NetEx/IP session services.....	9
Network Configuration Table (NCT).....	vii
network layer.....	7
normal termination.....	20
NRB error codes.....	147
driver service errors.....	150
general errors.....	148
host specific errors.....	150
license specific errors.....	150
network service errors.....	158
session service errors.....	155
transport service errors.....	153
NRBBLKI.....	32
NRBBLKO.....	32
NRBBUFA.....	29
NRBBUFL.....	29
NRBCLASS.....	32
NRBDMODE.....	29
NRBHOST.....	34

NRBIND.....	27	SCONFIRM.....	10
NRBLEN.....	27	SCONNECT.....	9
NRBMAXRT.....	32	SDISCONNECT.....	10
NRBNREF.....	29	segmenting.....	2
NRBOFFER.....	34	service WAIT options.....	22
NRBOSD.....	34	session data transfer.....	14
NRBPROTA.....	33	session layer.....	6
NRBPROTL.....	33	session layer requests.....	9
NRBREQ.....	28	session services.....	
NRBRESV1.....	34	abnormal termination.....	21
NRBRESV2.....	34	concurrent data transfer.....	16
NRBRESV3.....	34	data transfer.....	14
NRBSTAT.....	26	establishing a session.....	12
NRBTIME.....	31	normal termination.....	20
NRBUBIT.....	27	one-way data transfer.....	18
NRBUSER.....	34	read data transfer.....	14
one-way data transfer.....	18	terminating a session.....	20
Open Systems Interconnection (OSI).....	vii	write data transfer.....	14
OSI model.....	5	SOFFER.....	9
path.....	vii	SREAD.....	10
processor interface (PI).....	vii	SWAIT.....	10
programming notes.....	21	SWRITE.....	10
read data transfer.....	14	terminating a session.....	20
remote operator interface.....	2	transport layer.....	6
satellite communications.....	22	write data transfer.....	14
SCLOSE.....	10		